



INSTITUTO POLITÉCNICO
NACIONAL



ESCUELA SUPERIOR DE CÓMPUTO

TRABAJO TERMINAL II – 2016-B044

Máquinas de Turing y el problema de la universalidad como sistemas dinámicos discretos

Autores:

Juárez Martínez Sergio Eduardo
Manzano Mendoza César Iván

Director:

Dr. Genaro Juárez Martínez

Ing. en sistemas computacionales

1 de noviembre de 2017

Índice general

Índice de figuras	IV
Índice de cuadros	VI
Resumen	VII
1. Introducción	1
2. Estado del arte	3
2.1. Autómatas celulares y reconocedores conectados en una dirección	3
2.2. Autómatas móviles	4
2.3. Representación de las máquinas de Turing en dos dimensiones	5
2.4. Autómatas celulares universales pequeños	6
2.4.1. El juego de la vida	6
2.4.2. Capacidad de la regla 110 para simular una máquina de Turing	7
2.5. Autómatas celulares que simulan máquinas de Turing	7
2.5.1. La noción de computación	7
2.5.2. Espacios celulares simples computacionalmente universales	7
2.5.3. Simulación de autómatas celulares por autómatas celulares	8
2.5.4. Computación universal en autómatas celulares unidimensionales simples	8
2.6. Máquinas de Turing universales	9
3. Marco teórico	10
3.1. Teoría de autómatas y lenguajes formales	10
3.1.1. Autómatas	10
3.1.2. Máquinas de Turing	12
3.1.3. Universalidad	14
3.2. Sistemas dinámicos discretos	14
3.3. Autómatas Celulares	15
3.3.1. Clases de autómatas celulares	17
4. Diseño	20
4.1. Percepción de una dificultad	21
4.2. Identificación y definición de la dificultad	21
4.3. Soluciones propuestas para el problema: hipótesis	22
4.4. Verificación de la hipótesis mediante la acción	22

5. Desarrollo del Software	23
5.1. Máquinas de Turing	23
5.1.1. Opciones	26
5.1.2. Colours	27
5.1.3. Descripción de una máquina de Turing	29
6. Modelo de un autómatas celular computacionalmente universal	30
6.1. Análisis de una máquina de Turing para el balanceo de paréntesis	30
6.1.1. Descripción de la máquina	30
6.1.2. Representación de la cinta de la máquina de Turing	31
6.1.3. Construcción del autómatas	35
6.1.4. Representación de los movimientos a la derecha	35
6.1.5. Representación de los movimientos a la izquierda	36
6.1.6. Descripción completa de la función de transición del autómatas celular	37
6.1.7. Condición de paro en el autómatas	41
6.2. Algoritmo para el cálculo de un autómatas celular equivalente a una máquina de Turing arbitraria	41
6.3. Aplicación del algoritmo	42
6.3.1. Máquina para la suma binaria	42
6.3.2. Máquina que simula la regla 110	48
6.4. Reducción del número de símbolos auxiliares	51
6.5. Algoritmo para el cálculo de un autómatas celular con pocos estados auxiliares equivalente a una máquina de Turing arbitraria	52
6.6. Autómatas celular universal	53
6.6.1. Elección de la máquina de Turing universal que se simulará	53
6.6.2. Construcción del autómatas celular	55
7. Esbozo de una taxonomía inicial de las máquinas de Turing	57
7.1. El conjunto de máquinas de Turing a analizar	57
7.1.1. La máquina 0^n1^n	58
7.1.2. La máquina para el emparejamiento de paréntesis	59
7.1.3. La máquina duplicadora de unos	61
7.1.4. La máquina sumadora	62
7.1.5. La máquina para la resta unaria	63
7.1.6. La máquina de Turing universal con 7 estados y 4 símbolos	65
7.2. Esbozo de la taxonomía	66
8. Conclusiones	68

Índice de figuras

2.1. Reconocedor conectado en una dirección que reconoce las cadenas de paréntesis balanceados	4
2.2. Ejemplo de autómatas móvil [6]	5
2.3. Representación de una máquina de Turing en dos dimensiones[22]	6
2.4. Descripción de la regla 110. [6]	7
3.1. Grafo para el autómata elemental correspondiente a la regla 90 con 16 celdas en su espacio de evoluciones. Generado con Mathematica[12]	17
3.2. Grafo para el autómata elemental correspondiente a la regla 22 con 32 celdas en su espacio de evoluciones. Generado con DDLab[13]	17
3.3. Ejemplo del comportamiento del autómata elemental correspondiente a la regla 32. Pertenecer a la clase 1	18
3.4. Ejemplo del comportamiento del autómata elemental correspondiente a la regla 4. Pertenecer a la clase 2	18
3.5. Ejemplo del comportamiento del autómata elemental correspondiente a la regla 30. Pertenecer a la clase 3	19
3.6. Ejemplo del comportamiento del autómata elemental correspondiente a la regla 110. Pertenecer a la clase 4	19
5.1. Ventana inicial del programa	25
5.2. Gráfica de frecuencias de la máquina de Turing de emparejamiento de paréntesis con la entrada $((()))()$	27
5.3. Menú Options.	27
5.4. Menú Colours.	28
5.5. Ventana de la opción Edit del menú Colours con la máquina de Turing de emparejamiento de paréntesis.	28
5.6. Menú para elegir color	29
6.1. Diagrama de transiciones de la máquina de Turing para el emparejamiento de paréntesis	31
6.2. Dinámica en dos dimensiones de la máquina para la cadena de entrada $((()))()$	34
6.3. Dinámica en dos dimensiones de la máquina para el emparejamiento de paréntesis con la cadena $((((()))))((((()))())$	39
6.4. Dinámica en dos dimensiones del autómata para el emparejamiento de paréntesis con la cadena $a((((()))((((()))())$	40
6.5. Algoritmo para el cálculo de un autómata celular equivalente a una máquina de Turing arbitraria	41

6.6.	Dinámica en dos dimensiones de la máquina para el emparejamiento de paréntesis con la cadena $= 011100 + 010000 =$	46
6.7.	Dinámica en dos dimensiones del autómata para el emparejamiento de paréntesis con la cadena $a = 011100 + 010000 =$	47
6.8.	Dinámica en dos dimensiones de la máquina que simula el funcionamiento de la regla 110 con la cadena 010	49
6.9.	Dinámica en dos dimensiones de la máquina que simula el funcionamiento de la regla 110 con la cadena a010	50
6.10.	Algoritmo para el cálculo de un autómata celular con pocos estados equivalente a una máquina de Turing arbitraria	53
7.1.	Ejecución de la máquina 0^n1^n con un tamaño inicial de 5000 caracteres y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 63 generaciones.	58
7.2.	Ejecución de la máquina 0^n1^n con un tamaño inicial de 10 caracteres y con posición inicial del cabezal a la izquierda de la cinta de entrada. Esta cinta inicial alcanzó las 222 generaciones.	59
7.3.	Ejecución de la máquina para el emparejamiento de paréntesis con un tamaño inicial de 300 caracteres y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 938 generaciones.	60
7.4.	Ejecución de la máquina para el emparejamiento de paréntesis con una cadena inicial de 20 caracteres y con posición inicial del cabezal en el extremo izquierdo de la cinta de entrada. Esta cinta inicial alcanzó las 463 generaciones (más que en promedio de forma aleatoria)	61
7.5.	Ejecución de la máquina duplicadora de unos con una cadena inicial de 20 caracteres y con posición inicial del cabezal en el extremo izquierdo de la cinta de entrada. Esta cinta inicial alcanzó las 233 generaciones (más de el triple que el máximo obtenido en el experimento)	62
7.6.	Ejecución de la máquina sumadora con una cadena inicial de 1000 caracteres aleatorios y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 1001 generaciones.	63
7.7.	Ejecución de la máquina para la resta unaria con una cadena inicial de 1000 caracteres aleatorios y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 825 generaciones.	64
7.8.	Ejecución de la máquina para la resta unaria con una cadena inicial 31 caracteres arbitrarios el cabezal a la izquierda de la cadena de entrada. Esta cinta inicial alcanzó las 513 generaciones.	65
7.9.	Ejecución de la máquina de Turing universal con 7 estados y 4 símbolos con una cadena inicial 1000 caracteres con el cabezal en una posición inicial aleatoria sobre la cadena de entrada.	66

Índice de cuadros

6.1. Función de transición para la máquina de Turing para el emparejamiento de paréntesis	30
6.2. Lista de descripciones instantáneas al analizar la cadena $((()))()$	33
6.3. Transiciones del autómata celular que simulan el movimiento de la máquina de Turing a la derecha	35
6.4. Transiciones del autómata celular que simulan la transición $\delta\langle a, () = \langle a, (, R \rangle$ de la máquina de Turing	35
6.5. Evolución del autómata celular para los movimientos a la derecha	36
6.6. Transiciones del autómata celular para simular la transición $\delta\langle a, () = \langle b, X, L \rangle$ de la máquina de Turing	36
6.7. Evolución del autómata celular con movimientos a la izquierda	36
6.8. Descripción de la función de transición del autómata celular equivalente	37
6.9. Evolución del autómata para la cadena $((()))()$	38
6.10. Función de transición para la máquina de Turing que realiza una suma binaria	42
6.12. Descripción de la función de transición del autómata celular equivalente a la máquina que realiza una suma binaria	45
6.13. Función de transición para la máquina de Turing que simula la regla 110	48
6.14. Descripción de la función de transición del autómata celular equivalente a la máquina que simula la regla 110	48
6.15. Ejemplo de símbolos auxiliares con un comportamiento casi idéntico	51
6.16. Ejemplo de reglas ambiguas	51
6.17. Reglas para simular una transición de la forma $\delta(p, X) = (q, Y, L)$	52
6.18. Regla para desplazar el cabezal a la izquierda	52
6.19. Evolución del autómata celular para los movimientos a la derecha	52
6.20. Máquinas de Yuri Rogozhin	54
6.21. Función de transición para la máquina $UTM(7, 4)$	54
6.22. Función de transición para la máquina $UTM(5, 5)$	54
6.23. Función de transición para la máquina $UTM(4, 6)$	54
6.24. Estados necesarios para simular cada máquina	55
6.26. Descripción de la función de transición del autómata celular equivalente a la máquina que simula la regla 110	56

Resumen

En el presente documento se desarrolla la primera parte de una investigación que comprende el análisis de las máquinas de Turing y el fenómeno de la universalidad como sistemas dinámicos discretos. Con este propósito, buscamos crear una relación entre las máquinas de Turing y los autómatas celulares, de manera tal que estos últimos (los autómatas celulares) puedan comportarse como las máquinas de Turing. El trabajo terminal tiene como fines últimos la construcción de un autómata celular computacionalmente universal y el esbozo de una taxonomía que clasifique a las máquinas de Turing.

Los resultados que aquí se presentan incluyen el marco teórico, que contiene la definición formal de los conceptos necesarios para el desarrollo del trabajo; el estado del arte, en donde se exponen varios trabajos que se relacionan con nuestro proyecto o bien, que contienen resultados y observaciones que proporcionan un enfoque distinto al que nosotros hemos desarrollado para tratar conceptos semejantes; la descripción de la metodología que se utilizará, en donde se especifican las actividades que realizaremos a lo largo de esta investigación y finalmente los avances, en donde se muestra la documentación de las herramientas que hemos desarrollado y el análisis de una máquina de Turing clásica que nos ha llevado a la formulación de un algoritmo capaz de calcular un autómata celular equivalente a una máquina de Turing arbitraria.

Capítulo 1

Introducción

En 1936, Alan Mathison Turing desarrolló la conocida *máquina de Turing* como modelo para reconocer cualquier lenguaje formal a través de un procedimiento efectivo. Básicamente, este dispositivo es una máquina de estados finitos que dispone de una cinta de longitud infinita en la que se pueden leer y escribir datos. La máquina de Turing es lo suficientemente simple como para que podamos representar su configuración de manera precisa, utilizando una notación sencilla. La capacidad de estas máquinas para el reconocimiento de lenguajes formales les concede el mismo potencial para la resolución de problemas que una computadora actual, pues podemos plantear todos los problemas computables como una cuestión acerca de si un problema pertenece o no a un lenguaje determinado.

La teoría de los sistemas dinámicos describe en términos generales la forma en la que pueden cambiar los sistemas, qué tipos de comportamientos macroscópicos son posibles y qué tipo de predicciones sobre su evolución pueden realizarse, proporcionando un conjunto de herramientas para su estudio y análisis. Los sistemas dinámicos comprenden casi cualquier tipo de sistema imaginable, y pueden desarrollarse en tiempo y espacio continuos y discretos. Un muy claro ejemplo de este tipo de sistemas en tiempo y espacio discretos son los autómatas celulares.

Desde su creación hace más de 80 años, las máquinas de Turing han sido estudiadas a través de sus descripciones instantáneas (es decir, la configuración global de la máquina en un paso de tiempo específico durante el proceso en el que resuelve si una cadena pertenece o no a un lenguaje determinado). Es por ello que, en el presente Trabajo Terminal, se propone utilizar un nuevo enfoque para su estudio, concibiéndolas como sistemas dinámicos discretos.

El presente documento comprende el desarrollo y los resultados de una investigación cuyo punto de partida es la consideración de las máquinas de Turing como sistemas dinámicos discretos y cuyos objetivos se concentran en las hipótesis planteadas; *Existe un algoritmo tal que, tomando como entrada la descripción formal de una máquina de Turing arbitraria, devuelve como resultado un autómata celular capaz de simular a dicha máquina y Puede obtenerse un autómata celular universal a través de la aplicación del algoritmo anteriormente mencionado a una máquina de Turing universal.*

En primer lugar se abordó la problemática de la representación de las máquinas de Turing que, como fué mencionado con anterioridad, han sido estudiadas a través de sus descripciones instantáneas. Para ello se desarrolló una aplicación de escritorio capaz de desplegar la dinámica en dos dimensiones de cualquier máquina de Turing que lee una cadena de entrada. Dicha aplicación cuenta también con un conjunto de herramientas para el análisis de su comportamiento.

A continuación se analizó una máquina de Turing en particular, logrando obtener de ella

un autómata celular equivalente. Dicho autómata celular fue útil a su vez para desarrollar un algoritmo general, tal que, tomando como entrada la descripción formal de una máquina de Turing, produjera como salida la especificación de las reglas de un autómata celular equivalente a dicha máquina, comprobando la primera hipótesis.

El algoritmo general fue modificado para producir autómatas celulares más pequeños y posteriormente utilizado para obtener un autómata celular computacionalmente universal (comprobando así la segunda hipótesis). Para ello fueron consideradas las máquinas de Turing universales determinísticas de una cinta más pequeñas que se han encontrado. El análisis de dichas máquinas se incluye como parte de la investigación.

Capítulo 2

Estado del arte

2.1. Autómatas celulares y reconocedores conectados en una dirección

Los reconocedores conectados en una dirección son un modelo que utiliza los autómatas celulares para el reconocimiento de lenguajes. Un autómata celular conectado en una dirección es un autómata celular de una dimensión en el que el único vecino de una célula es la célula inmediatamente a su izquierda. Formalmente, una célula o celda conectada en una dirección, C , es una dupla $C = (Q, \delta)$, en donde Q es un conjunto finito no vacío de estados y $\delta : Q^2 \rightarrow Q$ es la función de transición. [7]

Un autómata celular conectado en una dirección Z es una tupla de tres elementos $(C, Q_I, \#)$, en donde $C = (Q, \delta)$ es una celda, $Q_I \subset Q$ es un conjunto de estados de entrada, y $\# \in Q_I$ es el símbolo especial que marca el límite. La función de transición está limitada de manera que $\delta(p, q) = \#$ si y sólo si $p = \#$, para una q arbitraria, tal que, $q \in Q$. [7]

Un reconocedor conectado en una dirección es un par $M = (Z, Q_A)$, en donde Z es un autómata celular conectado en una dirección y $Q_A \subset Q$ es un conjunto de estados de aceptación. M acepta una cadena $\omega \in (Q - \#)^*$ si M tiene una configuración inicial $\#^{\text{inf}} \omega \#^{\text{inf}}$, y en algún paso de tiempo, la celda más a la derecha en M diferente de $\#$, la celda de aceptación, entra en un estado dentro de Q_A . El conjunto de cadenas aceptadas por M define su lenguaje L . Es un problema abierto conocer si los reconocedores conectados en una dirección pueden reconocer los lenguajes libres de contexto [7].

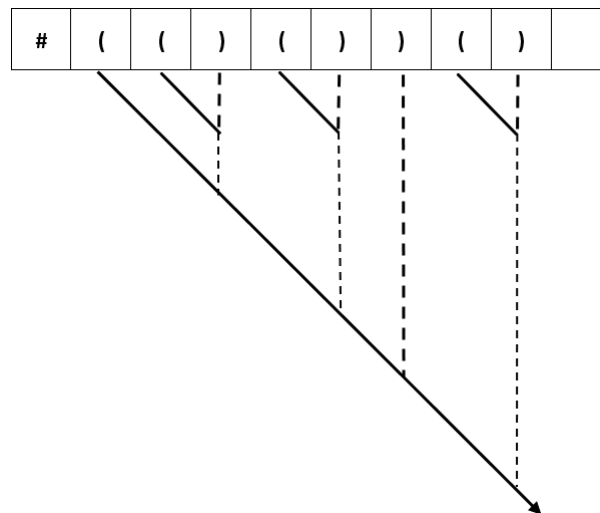


Figura 2.1: Reconocedor conectado en una dirección que reconoce las cadenas de paréntesis balanceados

2.2. Autómatas móviles

En su libro *A new kind of science*[6], publicado en 2002, Stephen Wolfram se pregunta qué tanto depende la capacidad de los autómatas celulares para generar comportamiento complejo de su característico procesamiento en paralelo. Para responder a esto, propone un modelo cuyo procesamiento se realiza una casilla a la vez, al que llama autómatas móviles. Los autómatas móviles son similares a los autómatas celulares, excepto que, en vez de actualizar todas sus celdas en paralelo, tienen una sola *celda activa* que se actualiza en cada iteración y que, como lo especifica una regla, se mueve de una posición a otra en el siguiente paso. La celda activa puede leer su estado y el de sus vecinos inmediatos.[6]

De entre varios miles de autómatas móviles, muy pocos presentan comportamiento complejo, por lo que Wolfram expone también un modelo generalizado de estos autómatas en el que una celda activa puede generar dos celdas activas y describe que el comportamiento complejo surge casi únicamente cuando hay muchas de estas, permitiendo concluir que el comportamiento complejo se encuentra con una frecuencia mucho mayor en modelos que se procesan en paralelo.

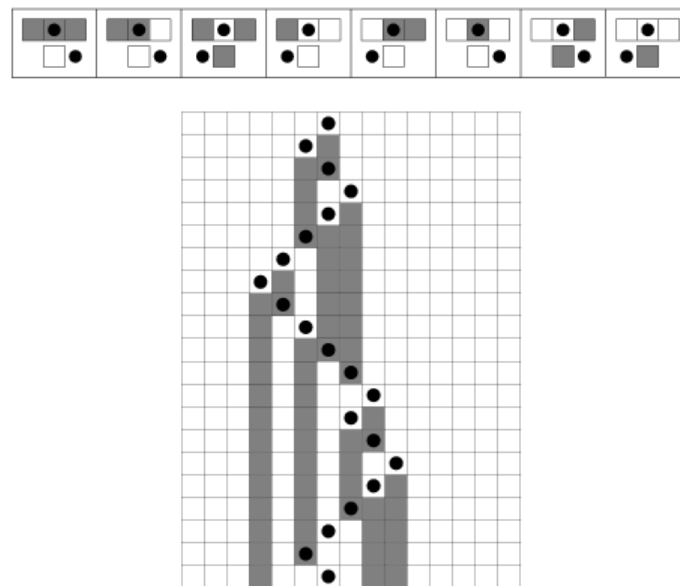


Figura 2.2: Ejemplo de autómatas móviles [6]

2.3. Representación de las máquinas de Turing en dos dimensiones

En la misma obra, Wolfram aborda las máquinas de Turing para analizar qué clase de comportamientos pueden generar. Para ello las representa en dos dimensiones tal y como lo hace con los autómatas móviles. La celda a la que apunta el cabezal de la máquina de Turing funge como la *celda activa* de un autómatas móviles. Además, existe un símbolo específico para cada estado de la máquina. Este se ilustra por encima de la celda a la que apunta.

Wolfram observa que las máquinas con dos y tres estados, por lo general, producen comportamientos bastante sencillos, pero un cuarto estado lleva a la generación de comportamiento más complejo. Sin embargo, el añadir aún más estados no aumenta el nivel de complejidad de las máquinas.

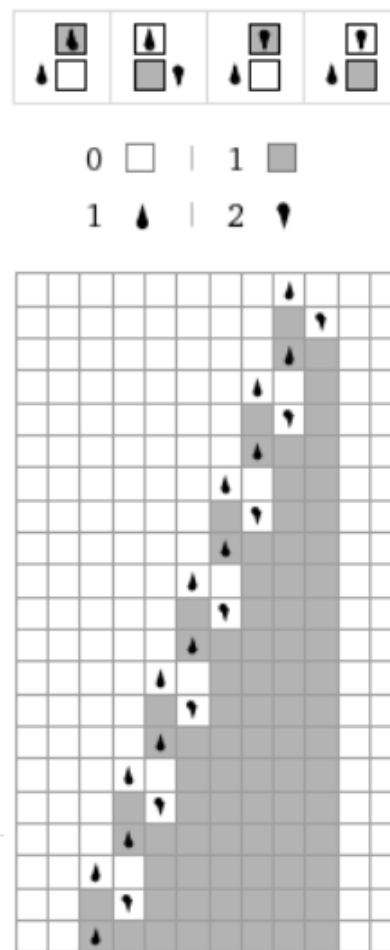


Figura 2.3: Representación de una máquina de Turing en dos dimensiones[22]

2.4. Autómatas celulares universales pequeños

2.4.1. El juego de la vida

En 1970, el matemático John Conway formuló un autómatas de dos estados (en dos dimensiones y con una vecindad de 8 vecinos) capaz de realizar computación universal. Éste lo llamó *juego de la vida* debido a la forma en la que enunció su regla de evolución.

Denotando las células en 1 como *vivas* y las células en 0 como *muertas*, Conway definió la regla en términos de procesos vitales: *nacimiento*, una célula muerta con exactamente tres vecinos vivos cobra vida en el siguiente paso de tiempo; *supervivencia*, una célula viva con dos o tres vecinos vivos se mantiene viva; *soledad*, una célula viva con menos de dos vecinos vivos muere y una célula muerta con menos de tres vecinos vivos permanece muerta y *sobrepoblación*, una célula viva o muerta con más de tres vecinos vivos muere o permanece muerta.

Conway formuló este autómatas celular mientras buscaba una regla que produjera comportamiento interesante (o quizá comportamiento que emulara al de los seres vivos).

Conway esbozó una prueba de que su autómatas celular podría simular una computadora universal. Es decir, que dada una configuración que codifique un programa y la entrada para el mismo, *El juego de la vida* o *Life* podría ejecutar dicho programa con esa entrada, produciendo

un patrón que representara la salida del programa. La prueba de Conway consistió en mostrar cómo algunas de las estructuras que se observan en el autómata pueden ajustarse para conformar las compuertas lógicas *and*, *or* y *not*, que combinadas pueden construir una computadora universal.

2.4.2. Capacidad de la regla 110 para simular una máquina de Turing

El autómata celular conocido como regla 110 de acuerdo con la numeración establecida por Wolfram en [6] es un autómata celular elemental que funciona de acuerdo con la regla que aparece en la figura 2.4. Un autómata celular elemental cuenta con un espacio de evoluciones de una sola dimensión infinito hacia ambas direcciones. Además, las células evolucionan de acuerdo con una función que las considera únicamente a sí mismas y a sus vecinos inmediatos. Es decir, la vecindad está formada de sólo tres elementos.

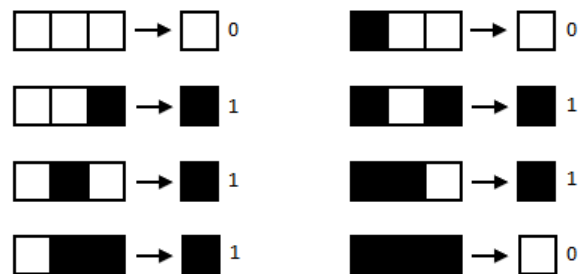


Figura 2.4: Descripción de la regla 110. [6]

En 1985, Wolfram conjeturó que la regla 110 era computacionalmente universal, pero esto no fue probado sino hasta 2004 en [9] por Matthew Cook. Para simular una máquina de Turing, este autómata celular tiene que simular primero un sistema Tag cíclico [8]. El sistema tag cíclico a su vez puede simular un sistema Tag tradicional, mismo que tiene la capacidad de simular una máquina de Turing como lo muestra Matthew Cook en [9].

2.5. Autómatas celulares que simulan máquinas de Turing

2.5.1. La noción de computación

En la sección 5 del capítulo 11 de su libro, *A new Kind of Science* [6], Wolfram muestra un conjunto de técnicas para construir un autómata celular a partir de una máquina de Turing. Si una máquina de Turing de s estados posee un alfabeto de k elementos, éste puede ser simulado por un autómata celular de $k(s + 1)$ colores.

2.5.2. Espacios celulares simples computacionalmente universales

En [19], Alvy Ray Smith explora el fenómeno de la universalidad en los espacios celulares unidimensionales, probando que basta un arreglo infinito de una sola dimensión para encontrar

ejemplos de universalidad. Smith afirma a su vez que la reducción de estados y de vecindades resulta particularmente sencilla para este modelo.

Smith utiliza como medida del tamaño de los autómatas el producto del número de estados en los que cada celda puede estar, p y la cantidad de vecinos que considera cada celda para evolucionar, q . Es decir, pq .

A lo largo de su publicación, Smith encuentra una gran cantidad de autómatas celulares universales con diferentes combinaciones de estados y vecinos, mismos que difieren a su vez en el tiempo que tardan en simular una máquina de Turing. El autómata más pequeño que Smith encuentra tiene un número de vecinos, $p = 9$ y un número de estados, $q = 4$, dando una medida $pq = 36$

2.5.3. Simulación de autómatas celulares por autómatas celulares

En [20], Jurgen Albert muestra que todo autómata celular con dos vecinos puede ser simulado por un autómata celular conectado en una sola dirección. También muestra que puede ser simulado por un autómata celular totalístico. Es decir que su evolución depende sólo de la suma de los valores de sus vecinos y no del estado específico de cada uno de estos.

Albert prueba también que se puede diseñar un autómata celular de 14 estados capaz de simular cualquier autómata celular totalístico conectado en una sola dirección con cualquier entrada. Dicho autómata, por consecuencia, puede simular cualquier autómata celular y por lo tanto es universal.

Por último, Albert considera los resultados obtenidos por Smith y plantea la existencia de autómatas celulares universales de 18 estados para máquinas de Turing universales de parejas de estados y símbolos (6, 6) y (7, 4). Dichos autómatas pueden ser utilizados para calcular un autómata celular universal, totalístico y conectado en una sola dirección.

2.5.4. Computación universal en autómatas celulares unidimensionales simples

En [21], Kristian Lindgren expone la construcción de un autómata celular universal unidimensional de siete estados y radio, $r = 1$. Para lograrlo, éste deja de utilizar cada símbolo de la cinta de la máquina de Turing y cada estado del cabezal como símbolos independientes en el autómata celular como había sucedido en los trabajos de Smith y Albert. En su lugar utiliza símbolos que representan tanto un estado como un símbolo de cinta. Para diferenciarlos, Lindgren utiliza uno y sólo un símbolo auxiliar que representa el movimiento del cabezal. El trabajo de Lindgren, a diferencia de los expuestos en las secciones anteriores, no puede generalizarse, pues como él mismo menciona, la construcción del autómata es un trabajo *nada trivial*. En el autómata celular diseñado en este trabajo, las representaciones de los símbolos que se encuentran en la cinta de la máquina están separados por un fondo periódico. Además, las configuraciones deben tener el espacio suficiente para que las partículas estáticas no entren en contacto mientras el cabezal las modifica.

2.6. Máquinas de Turing universales

Turing probó que puede construirse una máquina de Turing Universal que puede emular el funcionamiento de cualquier máquina de Turing. [1]

Claude Shannon mostró en 1956 que 2 colores eran más que suficientes para construir una máquina de Turing universal siempre y cuando se usara un número suficiente de estados [6]. Asimismo, demostró por reducción al absurdo que es imposible la existencia de una máquina de Turing Universal que sólo tenga un estado [15]. En 1962 Marvin Minsky propuso una máquina de Turing en un “sistema Tag”. Estos sistemas consisten en una secuencia de elementos, cada uno coloreado blanco o negro. Las reglas del sistema especifican en cada paso un número de elementos que deben ser removidos del inicio de la secuencia. Y luego, dependiendo de los colores de esos elementos, uno de estos posibles bloques es etiquetado al final de la secuencia. Poco se publicó sobre máquinas universales pequeñas después de los resultados de Minsky. Sin embargo, siguiendo la misma idea de simulación de sistemas Tag, entre los 80's y los 90's Yurii Rogozhin encontró ejemplos de máquinas universales con las combinaciones de números de estados y colores siguientes: $(24, 2)$, $(10, 3)$, $(7, 4)$, $(5, 5)$, $(4, 6)$, $(3, 10)$, y $(2, 18)$ [6].

Capítulo 3

Marco teórico

3.1. Teoría de autómatas y lenguajes formales

La teoría de autómatas es el estudio de dispositivos de cálculo abstractos, i.e. de las “máquinas”. En las décadas de los años cuarenta y cincuenta, una serie de investigadores estudiaron las máquinas más simples, las cuales todavía hoy denominamos “autómatas finitos”. Originalmente, estos autómatas se propusieron para modelar el funcionamiento del cerebro y, posteriormente, resultaron extremadamente útiles para muchos otros propósitos. [14]

Los autómatas son esenciales para el estudio de los límites de la computación y existen dos factores importantes a este respecto:

- ¿Qué puede hacer una computadora? (decidibilidad).
- ¿Qué puede hacer una computadora de manera eficiente? (intratabilidad).

3.1.1. Autómatas

Para definir formalmente un autómata, daremos antes una serie de conceptos fundamentales para su comprensión.

- Alfabeto (Σ). Es un conjunto de símbolos finito y no vacío. [14]
- Cadena de caracteres (palabra). Es una secuencia finita de símbolos seleccionados de algún alfabeto.
- Cadena vacía (ε). Es aquella que presenta cero apariciones de símbolos.
- Longitud de una cadena. Es el número de posiciones ocupadas por símbolos dentro de la cadena.
- Potencias de un alfabeto. Si Σ es un alfabeto, podemos expresar el conjunto de todas las cadenas de una determinada longitud de dicho alfabeto utilizando una notación exponencial. Definimos Σ^k para que sea el conjunto de las cadenas de longitud k , tales que cada uno de los símbolos de las mismas pertenece a Σ .
- Σ^* . Representa el conjunto de todas las cadenas de un alfabeto Σ .

- Σ^+ . El conjunto de cadenas no vacías del alfabeto Σ .
- Concatenación de cadenas. Sean x e y dos cadenas. Entonces, xy denota la concatenación de x e y .
- Lenguaje (L). Es un conjunto de cadenas, todas ellas seleccionadas de un Σ^* , donde Σ es un determinado alfabeto. Si Σ es un alfabeto y $L \subseteq \Sigma^*$, entonces L es un lenguaje de Σ .
- Problema. En la teoría de autómatas, es la cuestión de decidir si una determinada cadena es un elemento de un determinado lenguaje, i.e., dada una cadena w de Σ^* , decidir si w pertenece o no a L .

Autómatas finitos

El tipo de lenguajes conocidos como “lenguajes regulares” pueden describirse mediante un autómata finito. Un autómata finito tiene un conjunto de estados y su “control” pasa de un estado a otro en respuesta a las “entradas” externas. Una de las diferencias fundamentales entre las clases de autómatas finitos es si dicho control es “determinista”, lo que quiere decir que el autómata no puede encontrarse en más de un estado a un mismo tiempo, o “no determinista”, lo que significa que sí puede estar en varios estados a la vez. [14] A continuación se describirá la versión determinista, ya los lenguajes que reconocen son los mismos que la versión no determinista.

Autómatas finitos deterministas

El término “determinista” hace referencia al hecho de que para cada entrada sólo existe uno y sólo un estado al que el autómata puede hacer la transición a partir de su estado actual. Un autómata finito determinista puede representarse como una quintupla de la forma $A = (Q, \Sigma, \delta, q_0, F)$ en donde:

- Q es un conjunto finito de estados.
- Σ es un conjunto de símbolos de entrada.
- δ es una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado.
- q_0 es un estado inicial dentro de Q .
- F es un conjunto de estados finales o de aceptación.

Definimos el lenguaje del AFD como el conjunto de cadenas que acepta. Para saber si una cadena pertenece a dicho lenguaje, comenzaremos con el AFD en el estado inicial, q_0 . Consultamos la función de transición δ , por ejemplo $\delta(q_0, a_1) = q_1$ para hallar el estado al que pasará el AFD A después de procesar el primer símbolo de entrada a_1 . A continuación procesamos el siguiente símbolo de entrada, a_2 , evaluando $\delta(q_1, a_2)$; supongamos que este estado es q_2 . Continuamos aplicando el mismo procedimiento para hallar los estados $q_3, q_4, \dots, q_n$ tal que $\delta(q_{i-1}, a_i) = q_i$ para todo i . Si q_n pertenece a F , entonces la entrada $a_1a_2\dots a_n$ se acepta y, si no lo es se “rechaza”. [14]

3.1.2. Máquinas de Turing

En 1936, Alan Turing propuso la máquina de Turing como modelo de “cualquier posible computación”. Lo interesante es que todas las propuestas serias de modelos de computación tienen el mismo potencial; es decir, calculan las mismas funciones o reconocen los mismos lenguajes. La suposición no demostrada de que cualquier forma general de computación no permite calcular sólo las funciones recursivas parciales (o, lo que es lo mismo, que las máquinas de Turing o las computadoras actuales pueden calcular) se conoce como hipótesis de Church (por el experto en lógica A. Church) o tesis de Church.[14]

El modelo de la máquina de Turing consta de un dispositivo con una cinta cuadrículada y un cabezal que apunta a un elemento de la cinta. El cabezal puede leer y escribir en la cinta. La cinta contiene la cadena que se analiza e infinitos espacios en blanco a la izquierda y la derecha.[14]

Describimos una máquina de Turing mediante la siguiente séptupla: $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ en donde

- Q es el conjunto finito de estados de la unidad de control.
- Σ es el conjunto finito de símbolos de entrada.
- Γ es el conjunto completo de símbolos de cinta.
- δ es la función de transición. Los argumentos de $\delta(q, X)$ son un estado q y un símbolo de cinta X . El valor de $\delta(q, X)$ es de la forma (p, Y, D) en donde p es un estado, Y el símbolo que por el que se reemplaza X y por último D es la dirección en la que se mueve el cabezal.
- q_0 es el estado inicial.
- B es el símbolo de espacio en blanco.
- F es el conjunto de estados finales o de aceptación.

Descripciones instantáneas de la máquina de Turing

Para representar la configuración de una máquina de Turing, utilizamos una cadena en la que se muestran todos los elementos de la cinta entre el que está más a la izquierda y el que está más a la derecha que no son blancos. Si el cabezal apunta a una casilla en blanco, esta también se agrega. En la representación incluimos el cabezal. Por tanto, utilizaremos la cadena $X_1X_2...X_{i-1}qX_iX_{i+1}...X_n$ para representar una configuración en la que: [14]

- q es el estado de la máquina.
- El cabezal señala al símbolo i .

Al igual que con los autómatas de pila, utilizamos el símbolo $\vdash$ para representar la transición de la máquina de una configuración a otra y $\vdash^*$ para representar 0 o más transiciones.

Lenguaje de una máquina de Turing

Decimos que una cadena w pertenece al lenguaje de una máquina de Turing M si $q_0w \vdash^* \alpha p \beta$ para algún p en F ; es decir, si al analizar la cadena, se puede alcanzar un estado final. Decimos que la máquina para cuando llega a un estado q y un símbolo X tal que $\delta(q, X)$ no está definida. Por lo regular se considera que la máquina acepta una cadena si ésta se detiene estando en un estado final. Decimos que un lenguaje L es recursivamente numerable si $L = L(M)$ para alguna máquina de Turing M [14]

Lenguajes recursivos

Podemos clasificar a los lenguajes recursivamente enumerables dentro de dos clases:

- La primera clase corresponde a lo que tradicionalmente llamamos algoritmo. Para esta clase, existe una máquina de Turing que no sólo reconoce el lenguaje, sino que también informa de si una cadena de entrada no pertenece al mismo. Una máquina de Turing así siempre termina parándose, independientemente de si llega o no a alcanzar un estado de aceptación. A esta clase de lenguajes los llamamos lenguajes decidibles o lenguajes recursivos.
- La segunda clase está formada por aquellos lenguajes recursivamente enumerables que no son aceptados por ninguna máquina de Turing que garantice la parada en todos los casos. Estos lenguajes son aceptados de una forma poco adecuada; si la entrada pertenece al lenguaje, terminará aceptándola, pero si la entrada no pertenece al lenguaje, entonces la máquina de Turing continuará ejecutándose indefinidamente y nunca podremos estar seguros de que dicha entrada no será finalmente aceptada.

Máquina de Turing de varias cintas

Una máquina de Turing de varias cintas es parecida a la máquina de Turing tradicional pero posee un número finito de cintas, una única estación de control y una cabeza por cada cinta. La máquina se mueve conforme a los símbolos leídos por cada cabezal. Cada cabezal puede cambiar un símbolo de su cinta y moverse independientemente. [14]

Inicialmente:

- La entrada se coloca en la primera cinta.
- Toda las demás cintas sólo contienen símbolos en blanco.
- La unidad de control se encuentra en el estado inicial.
- Las cabezas de las demás cintas apuntan a una posición arbitraria.

Las máquinas de Turing con varias cintas pueden simular fácilmente a una máquina de Turing con una sola cinta. De la misma forma, una máquina de Turing con una sola cinta, puede simular a una máquina de múltiples cintas, por lo que reconocen el mismo tipo de lenguajes. [14]

Máquinas de Turing y computadoras

Es posible comparar la máquina de Turing con las computadoras que habitualmente usamos. Aunque estos modelos parecen bastante diferentes, pueden aceptar exactamente los mismos lenguajes: los lenguajes recursivamente enumerables. Dado que no hemos definido matemáticamente el concepto de “computadora común”, los argumentos aplicados en esta sección son necesariamente informales. [14]

En principio, es posible simular una máquina de Turing mediante una computadora real si aceptamos que existe un suministro potencialmente infinito de un dispositivo de almacenamiento extraíble, como por ejemplo, un disco, para simular la parte en que no hay espacios en blanco de la cinta de la MT. Puesto que los recursos físicos con los que se construyen los discos no son infinitos, este argumento es cuestionable. Sin embargo, dado que la cantidad de dispositivos de almacenamiento que existe en el universo es desconocida y, si duda, muy grande, la suposición de que existen recursos infinitos, como en una cinta de una MT, es realista en la práctica y generalmente se acepta. [14]

Una máquina de Turing puede simular el almacenamiento y la unidad de control de una computadora real empleando una cinta para almacenar todas las posiciones y los contenidos correspondientes a los registros, la memoria principal, los discos y otros dispositivos de almacenamiento. Por tanto, podemos estar seguro de que algo que una máquina de Turing no pueda hacer, tampoco lo podrá hacer una computadora real. [14]

3.1.3. Universalidad

El remarcable descubrimiento que permitió la revolución de las computadoras es que es posible construir sistemas universales cuya construcción permanece fija, pero que pueden ser programadas para realizar diversas actividades. De hecho, esta es la forma en la que trabajan las computadoras prácticas: el hardware permanece fijo mientras que la computadora puede ser programada para diferentes tareas al cargar diferentes componentes de software. La idea de universalidad es también la base de los lenguajes de programación. Para cada lenguaje hay un conjunto específico de operaciones primitivas que son encadenadas juntas de diferentes maneras para crear programas que resuelven tareas distintas. Las características específicas de un sistema de computación o de un lenguaje de programación afectarán indudablemente qué tan fácil es realizar una tarea determinada [6]. Lo interesante es que todas las propuestas serias de modelos de computación tienen el mismo potencial; es decir, calculan las mismas funciones o reconocen los mismos lenguajes. La suposición no demostrada de que cualquier forma general de computación no permite calcular sólo las funciones recursivas parciales (o, lo que es lo mismo, que las máquinas de Turing o las computadoras actuales pueden calcular) se conoce como hipótesis de Church (por el experto en lógica A. Church) o tesis de Church [14].

El punto básico es que, si un sistema es universal, entonces éste es capaz de emular cualquier otro sistema, y como resultado este debe de ser capaz de producir comportamiento tan complejo como el comportamiento de cualquier otro sistema. [6]

3.2. Sistemas dinámicos discretos

La teoría de los sistemas dinámicos describe, en términos generales, en qué formas los sistemas pueden cambiar, qué tipos de comportamiento macroscópico son posibles y qué tipo

de predicciones sobre el comportamiento se pueden hacer. La palabra *dinámica* significa cambio, y los sistemas dinámicos, en general, son sistemas que pueden cambiar a lo largo del tiempo de alguna forma. Los sistemas dinámicos incluyen casi cualquier tipo de sistemas imaginables: los sistemas planetarios, el sistema nervioso, el cambio climático, la bolsa de valores e inclusive las rocas. Los sistemas dinámicos han estado en voga en la ciencia popular debido a los resultados fascinantes que provienen de una de sus consecuencias, el estudio del caos [1].

Un sistema dinámico es una especie de modelo matemático que busca capturar formalmente la noción intuitiva de un sistema determinista arbitrario, ya sea este reversible o no reversible, con tiempo y espacio discretos o continuos [2].

Los sistemas dinámicos discretos, por su parte, son esencialmente funciones iterativas. Un sistema dinámico discreto puede ser caracterizado como una función que se compone de sí misma una y otra vez. Por ejemplo, consideremos la función $f(x) = -x^3$. Si componemos f consigo misma misma, obtenemos

$$f^2(x) = (f \circ f)(x) = -(-x^3)^3 = x^9$$

Iterando este proceso obtenemos

$$\begin{aligned} f^3(x) &= (f \circ f \circ f)(x) = (f \circ f^2)(x) = -(x^9)^3 = -x^{27} \\ f^4(x) &= (f \circ f \circ f \circ f)(x) = (f \circ f^3)(x) = -(-x^{27})^3 = x^{81} \\ &\vdots \\ f^n(x) &= (f \circ f^{n-1})(x) = (-1)^n x^{3^n} \end{aligned}$$

En donde n es un número natural. [3]

3.3. Autómatas Celulares

Un ejemplo de sistemas dinámicos discretos son los autómatas celulares. Los autómatas celulares son modelos prototípicos para sistemas y procesos complejos, conformados por un gran número de componentes simples, idénticos y conectados de forma local. El estudio de estos sistemas ha generado gran interés a través de los años por su habilidad para generar un amplio espectro de patrones muy complejos haciendo uso de conjuntos de reglas relativamente simples. Además, parecen capturar muchas características esenciales del comportamiento auto-organizado de sistemas reales [4].

A pesar de su muy sencilla construcción, nada parecido a los autómatas celulares parece haber sido considerado antes de la década de 1950. Sin embargo, a lo largo de la misma (inspirada de varias formas por el advenimiento de las computadoras electrónicas), varios tipos de sistemas equivalentes a los autómatas celulares fueron introducidos de forma independiente. La forma más conocida en la que se introdujeron los autómatas celulares (y la que eventualmente condujo a nombrarlos de tal forma) es a través del trabajo de John von Neumann, que buscaba desarrollar un modelo abstracto de autorreproducción biológica (tema que emergió de las investigaciones en cibernética alrededor de 1947). Von Neumann empezó pensando en modelos en 3 dimensiones descritos por ecuaciones diferenciales. A la brevedad, dejó de pensar en la robótica e imaginó la implementación de un ejemplo usando un set de construcción de

juguete. Por analogía con los circuitos eléctricos, se dio cuenta de que bastarían dos dimensiones. Y siguiendo una sugerencia de Stanislaw Ulam en 1951 (quien pudo ya haber considerado el tema de forma independiente), simplificó el modelo dando paso a un autómata celular en dos dimensiones. El autómata que construyó en 1953 constaba de 29 estados para cada celda y un complicado conjunto de reglas para emular las operaciones de los componentes de una computadora electrónica y varios dispositivos mecánicos [6].

Formalmente, un autómata celular es una tupla $\langle \mathbf{L}, \mathbf{Q}, u, f \rangle$ formada por un arreglo multi-dimensional uniforme $\mathbf{L}$ de autómatas finitos, un conjunto finito de estados $\mathbf{Q}$, una conexión local entre los autómatas o vecindad $u, u : \mathbf{L} \rightarrow \mathbf{L}^k$, k es un entero positivo, y una función de transición sobre los autómatas $f, f : \mathbf{Q}^k \rightarrow \mathbf{Q}$. Los autómatas del arreglo actualizan su estado de forma simultánea y en tiempo discreto. Cada autómata x dentro del arreglo $\mathbf{L}$ calcula su siguiente estado haciendo uso de la regla $x^{t+1} = f(u(x)^t)$, en donde x^{t+1} es el estado del autómata x en el paso de tiempo $t + 1$ y $u(x)^t$ es el estado de la vecindad del autómata $u(x)$ en el paso de tiempo t [5].

En la mayoría de los casos, para definir la vecindad $u(x)$ de una celda x es suficiente determinar cuáles son las celdas que se encuentran a una distancia no mayor a r usando una métrica M , en donde r es un entero positivo. Formalmente

$$u(x) = \{y \in \mathbf{L} : |x - y|_M \leq r\}$$

[5]

La configuración de un autómata celular es un mapeo $c : \mathbf{L} \rightarrow \mathbf{Q}$, que asigna un estado de $\mathbf{Q}$ a cada celda del arreglo $\mathbf{L}$. Una función de transición global es un mapeo $G : \mathbf{Q}^{\mathbf{L}} \rightarrow \mathbf{Q}^{\mathbf{L}}$ que transforma cualquier configuración c a otra configuración $c', c' = G(c)$ aplicando la función de transición local f a la vecindad de cada celda en el arreglo $\mathbf{L}$. Decimos que un autómata celular es determinístico si para cualquier configuración c existe una única configuración c' tal que $G(c) = c'$. Dada la configuración inicial c^0 , representamos la evolución del autómata como se muestra a continuación:

$$c^0 \rightarrow c^1 \rightarrow \dots c^t \rightarrow c^{t+1} \rightarrow \dots c^{t+p} = x^t$$

Eventualmente el autómata celular cae en un ciclo con período $p > 0$. Si $p = 1$, el autómata se estaciona en un estado fijo [5].

La dinámica global de los autómatas celulares finitos puede visualizarse en un grafo de sus transiciones globales. Cada nodo del grafo corresponde a una configuración única del autómata celular y los arcos que conectan los nodos representan las transiciones globales entre configuraciones [5].

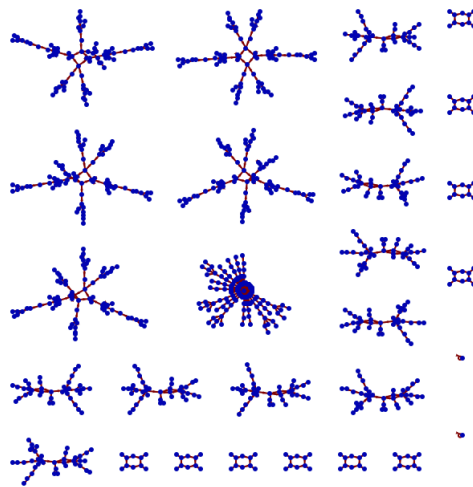


Figura 3.1: Grafo para el autómata elemental correspondiente a la regla 90 con 16 celdas en su espacio de evoluciones. Generado con Mathematica[12]

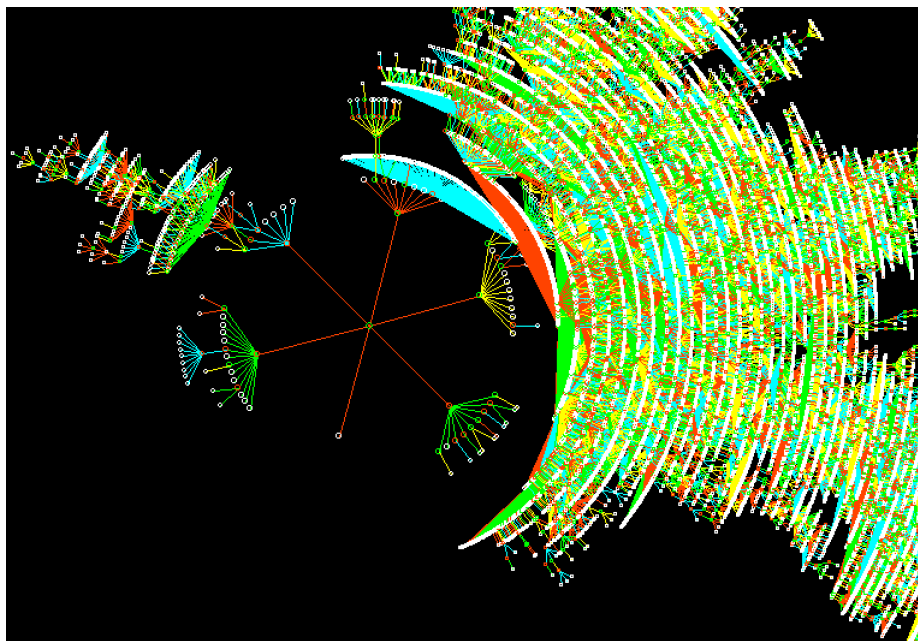


Figura 3.2: Grafo para el autómata elemental correspondiente a la regla 22 con 32 celdas en su espacio de evoluciones. Generado con DDLab[13]

3.3.1. Clases de autómatas celulares

En su libro, *A new kind of science*[6], publicado en 2002, Wolfram define cuatro clases dentro de las cuales, afirma, pueden ser fácilmente clasificados todo tipo de autómatas celulares según los patrones que de estos emergen para condiciones iniciales arbitrarias. Las clases están convenientemente enumeradas según el incremento en su complejidad y cada una tiene ciertas características distintivas inmediatas [6].

En la clase 1, el comportamiento es muy sencillo, y casi todas las configuraciones iniciales llevan exactamente al mismo estado uniforme.

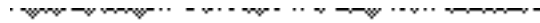


Figura 3.3: Ejemplo del comportamiento del autómata elemental correspondiente a la regla 32. Pertenecer a la clase 1

En la clase 2, hay muchos posibles estados finales diferentes, pero todos ellos consisten de ciertos conjuntos de estructuras simples que permanecen estáticas o se repiten después de unos cuantos pasos.

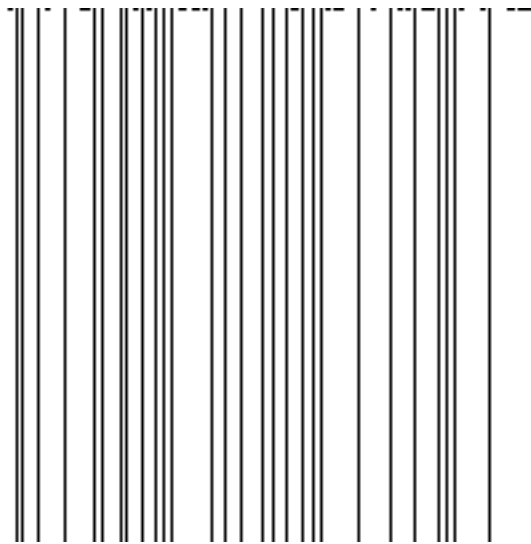


Figura 3.4: Ejemplo del comportamiento del autómata elemental correspondiente a la regla 4. Pertenecer a la clase 2

En la clase 3, el comportamiento es más complicado, y parece en muchos sentidos aleatorio, a pesar de que triángulos y otras figuras a pequeña escala son vistas casi siempre a cierto nivel.

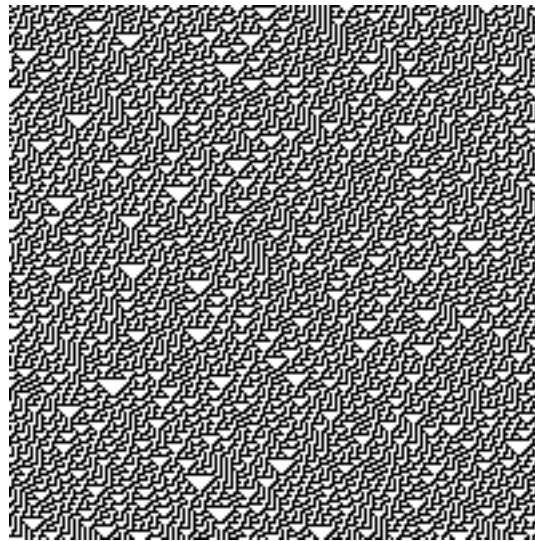


Figura 3.5: Ejemplo del comportamiento del autómata elemental correspondiente a la regla 30. Pertenecce a la clase 3

La clase 4 involucra una mezcla de orden y aleatoriedad: aparecen estructuras específicas que por sí mismas son bastante simples, pero dichas estructuras se van moviendo e interactuando las unas con las otras en formas muy complejas.

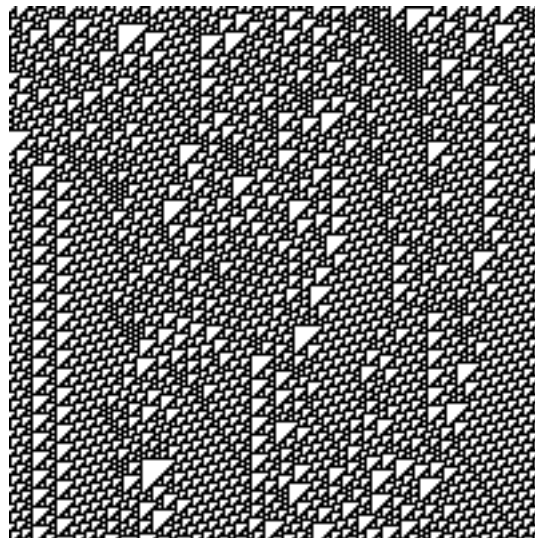


Figura 3.6: Ejemplo del comportamiento del autómata elemental correspondiente a la regla 110. Pertenecce a la clase 4

Capítulo 4

Diseño

El presente trabajo es una investigación en el área de las ciencias de la computación, por lo que se utilizará el método científico como metodología.

El método científico es un conjunto de procedimientos por los cuales se plantean los problemas científicos y se ponen a prueba las hipótesis y los instrumentos de trabajo investigativo [16].

Es importante recalcar que lo que importa y es fundamental en el método científico no es el descubrimiento de verdades en todo momento, sino más bien el determinar cuál ha sido el procedimiento para demostrar que un enunciado es así, pues cada ciencia plantea y requiere de un método especial, según sea la naturaleza de los hechos que estudia, pero los pasos que se han de dar o seguir están regulados por el método científico [16].

En el método científico se conjugan la inducción y la deducción, es decir se da el pensamiento reflexivo. En el proceso del pensar reflexivo se dan 5 etapas para resolver un problema.

1. Percepción de una dificultad. El individuo encuentra algún problema que le preocupa y se halla sin los medios para llegar al fin deseado.
2. Identificación y definición de la dificultad. El individuo efectúa observaciones que le permiten definir su dificultad con mayor precisión.
3. Soluciones propuestas para el problema: hipótesis. A partir del estudio de los hechos, el individuo formula conjeturas acerca de las posibles soluciones del problema, esto es, formula una hipótesis. Una hipótesis es una proposición que puede ser puesta a prueba para determinar su validez. Siempre lleva a una prueba empírica; es una pregunta formulada de tal modo que se puede prever una respuesta de alguna especie.
4. Deducción de las consecuencias de las soluciones propuestas. El individuo llega a la conclusión de que, si cada hipótesis es verdadera, le seguirán ciertas consecuencias.
5. Verificación de la hipótesis mediante la acción. El individuo pone a prueba cada una de las hipótesis, buscando hechos observables que permitan confirmar si las consecuencias que deberían seguir se producen o no. Con este procedimiento puede determinar cuál de las hipótesis concuerda con los hechos observables, y así hallar la solución más confiable para su problema. [16]

Siguiendo este método tenemos que cumplir con las cinco etapas, cuyas actividades realizadas y a realizar estarán descritas en las siguientes secciones.

4.1. Percepción de una dificultad

El estudio de las máquinas de Turing está centrado en la historia de sus descripciones instantáneas, lo cuál resulta un tanto complicado cuando éstas se presentan en gran cantidad. Es por ello que se plantea como dificultad y se propone indagar otras maneras de representar y analizar las máquinas de Turing. Específicamente como un sistema dinámico discreto y concretamente como un autómata celular.

De manera más precisa se puede especificar nuestro objetivo como:

Analizar las máquinas de Turing y el problema de la universalidad como sistemas dinámicos discretos.

Y nuestros objetivos específicos como:

- Desarrollar una aplicación capaz de desplegar la dinámica en dos dimensiones de cualquier máquina de Turing dadas su descripción formal y una cinta para ser procesada.
- Interpretar el comportamiento de la dinámica espacial de algunas máquinas de Turing clásicas haciendo uso de la aplicación antes mencionada.
- Esbozar una taxonomía inicial de las máquinas de Turing.
- Construir el modelo de un autómata celular computacionalmente universal.
- Presentar los resultados de la investigación en un documento.

Cabe destacar que esta parte del trabajo fue realizada durante el diseño del protocolo para este trabajo.

4.2. Identificación y definición de la dificultad

En esta parte de la investigación se realizó la búsqueda y comprensión de la información relacionada con el proyecto. De igual forma, se definieron de manera formal los conceptos fundamentales para la elaboración de nuestro análisis.

Dichos conceptos y tópicos han sido plasmados en el presente documento como parte del marco teórico y el estado del arte. En estas secciones se incluyen conceptos como teoría de autómatas, máquinas de Turing, sistemas dinámicos, autómatas celulares; así como trabajos relacionados como los desarrollados por Wolfram en el área de autómatas celulares, autómatas móviles y el uso de la representación en dos dimensiones de una máquina de Turing. También se incluyen artículos dedicados a analizar las capacidades de computación de los autómatas celulares.

Asimismo para examinar el comportamiento de diferentes máquinas de Turing y cumpliendo con uno de nuestros objetivos específicos, se desarrolló una aplicación capaz de desplegar la dinámica en dos dimensiones de cualquier máquina de Turing dadas su descripción formal y una cinta para ser procesada, la cual será descrita con mayor detalle en los avances realizados.

4.3. Soluciones propuestas para el problema: hipótesis

Una vez desarrollado el software y habiendo realizado las observaciones pertinentes, pudimos formular algunas hipótesis, entre las cuáles destacan:

1. Existe un algoritmo tal que, tomando como entrada la descripción formal de una máquina de Turing arbitraria, devuelve como resultado un autómata celular capaz de simular a dicha máquina.
2. Puede obtenerse un autómata celular universal a través de la aplicación del algoritmo anteriormente mencionado a una máquina de Turing universal.

4.4. Verificación de la hipótesis mediante la acción

En el presente documento se ha buscado verificar la primera hipótesis enlistada a través de la formulación de un algoritmo que cumpla con las características descritas. Se ha llegado a un algoritmo capaz de encontrar los autómatas celulares equivalentes a algunas máquinas de Turing específicas. Para comprobar la hipótesis es menester demostrar que este algoritmo funciona para cualquier máquina de Turing. el desarrollo de esta demostración se llevará a cabo en la segunda parte de este trabajo terminal. Dicho algoritmo es presentado en la sección de avances realizados.

La verificación de la segunda hipótesis se ha relegado a la segunda parte del trabajo debido a la necesidad de atención de nuestros resultados con la primer hipótesis.

Capítulo 5

Desarrollo del Software

Como se ha mencionado, para la etapa de observación fue necesario el desarrollo de una herramienta que mostrara la dinámica de una máquina de Turing en dos dimensiones. Al ser ésta una herramienta importante para el desarrollo de la investigación, pero no siendo un producto final, la documentación de la misma se ha dejado a un lado y nos enfocaremos en mostrar las capacidades e instrucciones básicas de uso.

El desarrollo de la aplicación se realizó por prototipos, añadiendo funcionalidades requeridas por nuestro director el Dr. Genaro Juárez Martínez o requeridas por nosotros para mayor comodidad al momento de realizar las observaciones.

Esta aplicación está desarrollada en el lenguaje de programación C++ y específicamente con el framework Qt. Esto debido a que el framework permite el desarrollo de aplicaciones con interfaz gráfica de usuario de manera sencilla y a la capacidad de C++ en cuanto a velocidad y facilidad de desarrollo.[17] A continuación se describirán los módulos con los que cuenta el software.

5.1. Máquinas de Turing

El programa cuenta con un panel en donde se muestran las diferentes acciones que se pueden realizar, a continuación se enlistan los elementos de esta ventana con respecto a la figura 5.1.

1. Éste es un combo box, en el cuál se selecciona la máquina de Turing con la cuál se desea trabajar. Las máquinas se cargan de manera previa haciendo uso de su definición formal, de forma semejante a como se describió en el marco teórico. El archivo en el que se especifica dicha descripción será abordado con más detalle posteriormente.
2. Es una barra que sirve para aumentar el zoom de la representación de la máquina de Turing en dos dimensiones.
3. Es un check box que, al estar seleccionado y ejecutar una máquina de Turing con una cinta de entrada, muestra la evolución de la máquina de manera tradicional (como es descrito en el marco teórico) en el cuadro 12.
4. Es un check box que indica, al estar seleccionado, que al generar una cinta de entrada aleatoria, ésta se mostrará en la sección 11.

5. Es un check box que indica, al estar seleccionado, que se mostrará el cabezal de la máquina en la imagen generada.
6. Es un check box que indica, al estar seleccionado y generar una cinta de entrada aleatoria, que dicha cinta debe ser aceptada por la máquina de Turing.
7. Botón que al ser oprimido genera una cinta de entrada aleatoria con los valores especificados en 4, 6 y 8. Una vez generada esta cinta, es ejecutada la máquina de Turing previamente seleccionada en 1 con los valores especificados en 3, 5, 9, 10, 13 y 14.
8. Caja en la cuál se especifica la longitud de la cadena aleatoria a generarse. Éste valor es mandado como parámetro a un programa especificado junto con la descripción de la máquina de Turing.
9. Check box que indica, al estar seleccionado, que el cabezal se colocará de forma inicial en una posición aleatoria en el rango de la longitud de la cinta de entrada.
10. Caja en la cuál se especifica la posición inicial del cabezal al momento de ejecutar la máquina de Turing, esto tomando como referencia 0 que significa poner el cabezal a la izquierda del primer símbolo de la cadena, los números positivos mover el cabezal ese número de veces a la derecha y números negativos mover el cabezal ese número de veces a la izquierda.
11. Cuadro de texto en el cuál se introduce la cinta de entrada a ser procesada por la máquina de Turing y en el que se imprime la cadena generada de manera aleatoria en tal caso.
12. Cuadro de texto en el cuál se imprimen las descripciones instantáneas en caso de ser requerido.
13. Caja en la cuál se indica la equivalencia de pixeles cuadrados para los símbolos de cinta. En caso de ser 1, se utiliza 1 pixel por símbolo de cinta; en caso de ser 2, se utilizan 4 pixeles por cada símbolo de cinta, etc.
14. Caja en la cuál se indica el grosor del borde de cada cuadro que representa un símbolo de cinta.
15. Botón que al ser presionado ejecuta la máquina de Turing actualmente seleccionada en 1 con los valores especificados en 3, 5, 9, 10, 13 y 14 sobre la cinta actual en el cuadro de texto 11.

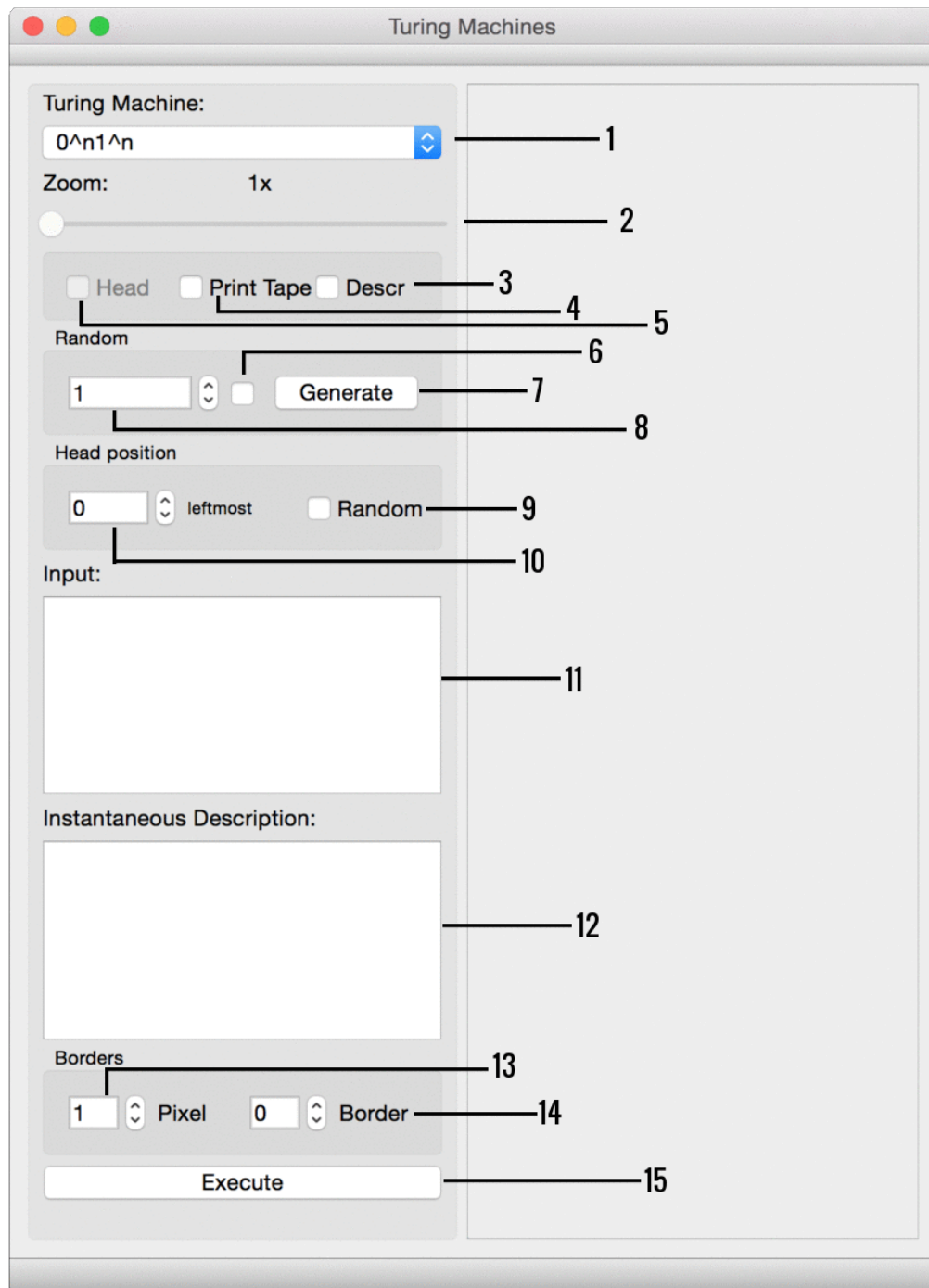


Figura 5.1: Ventana inicial del programa

Aunado a esto se disponen con dos opciones en la barra de menús: Options y Colours. Estos se describirán a continuación.

5.1.1. Opciones

En esta sección se dispone de varias opciones que facilitan el análisis y el compartir algunos de los resultados obtenidos, tales como la ejecución de una máquina de Turing, una cinta generada aleatoriamente, la lista de descripciones instantáneas o el despliegue de una gráfica de frecuencias de cada símbolo respecto al tiempo. A continuación se describirán estos elementos con mayor detalle respecto a la figura 5.3.

- Load Tape. Permite cargar una cinta de entrada. Este archivo debe tener la terminación ".cinta".
- Save Tape. Permite guardar la cinta actual con la que se está trabajando.
- Save image. Permite guardar la imagen de la ejecución actual de la máquina de Turing en dos dimensiones. Se puede guardar en formato ".bmp", ".jpgz ".png".
- Save Description. Permite guardar la descripción instantánea actual que se encuentra en el cuadro de texto especificado para ello.
- Symbols graph. Despliega una gráfica de frecuencias de los símbolos contra el tiempo, la cuál puede ser guardada en formato de imagen ".bmp", ".jpgz ".png". Un ejemplo de ello se muestra en la figura 5.2

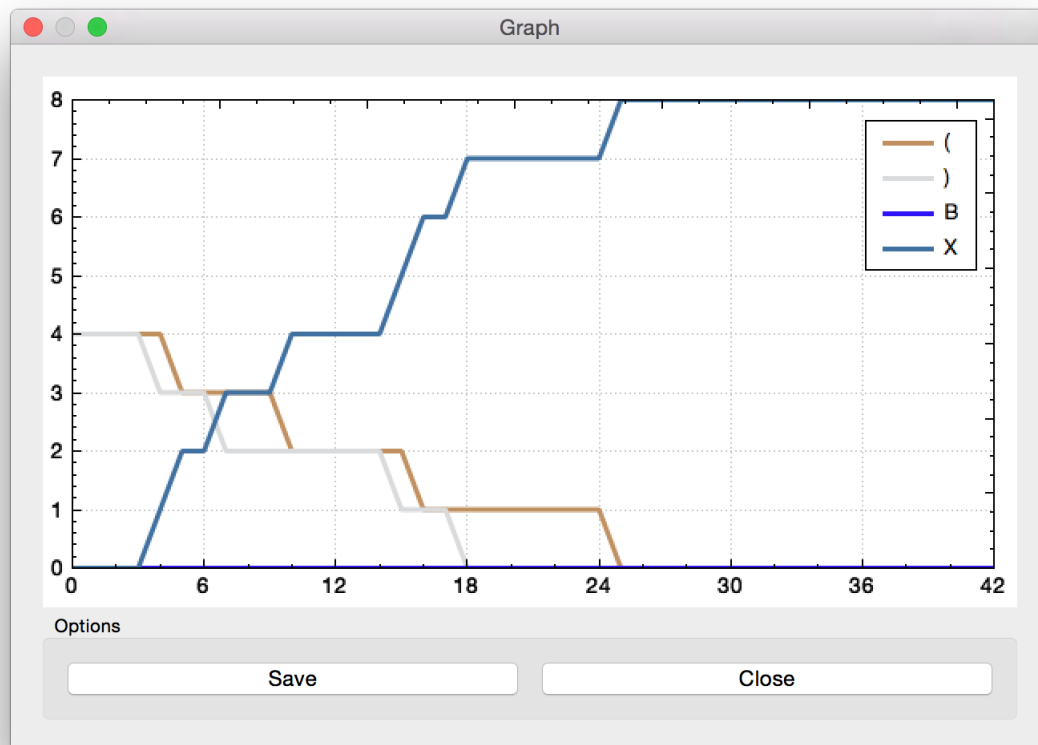


Figura 5.2: Gráfica de frecuencias de la máquina de Turing de emparejamiento de paréntesis con la entrada $((()))$

- Switch to Automaton. Abre la ventana en la que se dispone de la ejecución de los autómatas celulares equivalentes a las máquinas de Turing disponibles.

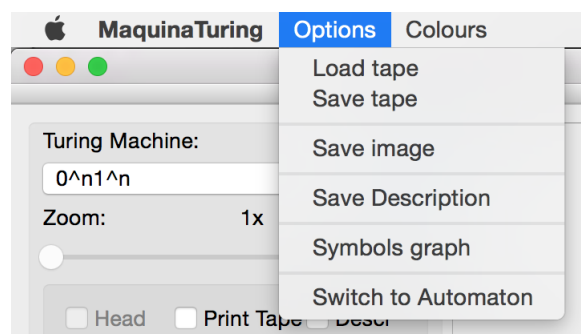


Figura 5.3: Menú Options.

5.1.2. Colours

Para facilitar el análisis de la representación de las máquinas de Turing en dos dimensiones se agregó la opción Colores, el cuál sólo incluye la opción Edit como se muestra en la figura 5.4

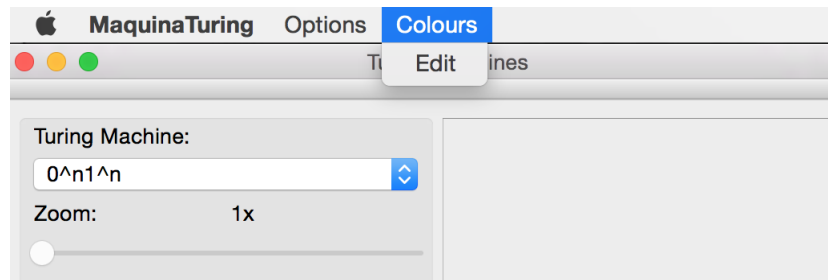


Figura 5.4: Menú Colours.

Al hacer clic en la opción Edit del menú Colours se despliega una ventana en la que hay una tabla con los símbolos de la máquina de Turing actualmente seleccionada y los colores actuales como la que se muestra en la figura 5.5.

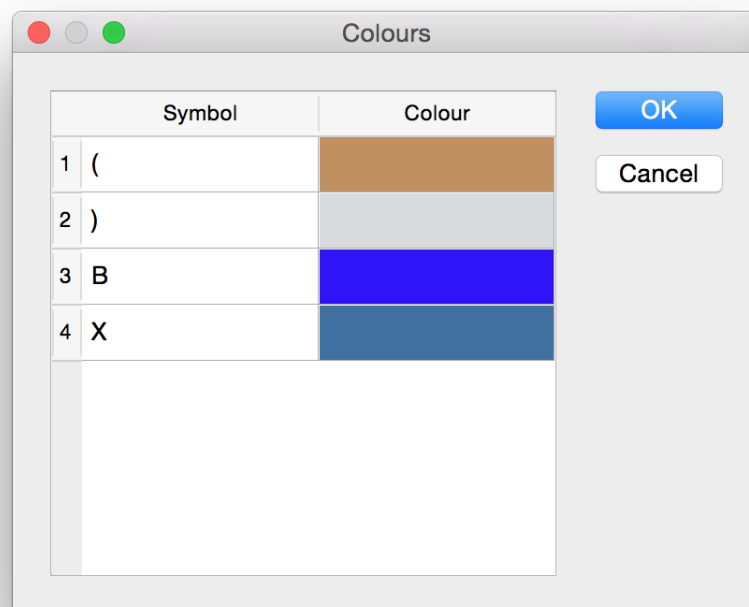


Figura 5.5: Ventana de la opción Edit del menú Colours con la máquina de Turing de emparejamiento de paréntesis.

Al hacer clic en la celda de alguno de los símbolos de la tabla se muestra una ventana como la de la figura 5.6 en la que se debe seleccionar el nuevo color para el símbolo deseado y al dar clic en "OK" ese será el color para todas las ejecuciones con esa máquina de Turing.

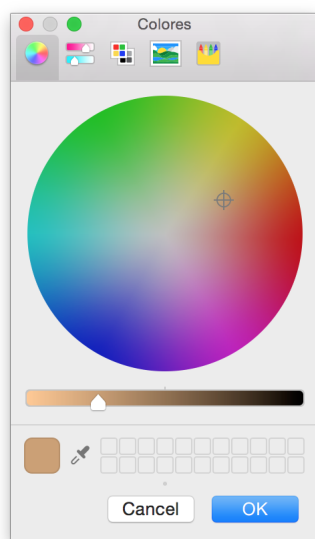


Figura 5.6: Menú para elegir color

5.1.3. Descripción de una máquina de Turing

A continuación se describirá cómo está conformado un archivo en el que se describe una máquina de Turing para que la lea el programa. Un ejemplo del archivo se muestra en las siguientes líneas, esta máquina de Turing es la de emparejamiento de paréntesis.

```

1 Paréntesis
2 a
3 B
4 d
5 ),a,X,b,i;( ,a,( ,a,d;X,a,X,a,d;B,a,B,c,i;( ,b,X,a,d;X,b,X,b,i;),c,) ,e ,
   d;( ,c,( ,e,d;X,c,X,c,i;B,c,B,d,i;
6 (;);
```

La primer línea es el nombre de la máquina de Turing que se mostrará. La segunda es el estado inicial. La tercera el símbolo blanco. La cuarta los estados de aceptación separados por una coma. La quinta línea son las transiciones válidas separadas por “;”. Cada una de las transiciones contiene los siguientes elementos separados por una coma: Símbolo de cinta actual, estado actual, siguiente símbolo de cinta, siguiente estado. Finalmente se enlistan los símbolos de entrada válidos para generar cintas de entrada aleatorias.

Capítulo 6

Modelo de un autómeta celular computacionalmente universal

6.1. Análisis de una máquina de Turing para el balanceo de paréntesis

La presente sección describe cómo se analizó una máquina de Turing capaz de reconocer cadenas de paréntesis balanceados para crear un autómeta celular que emulara su comportamiento en dos dimensiones.

6.1.1. Descripción de la máquina

Consideremos la máquina $M = (Q_M, \Sigma_M, \Gamma, \delta, q_0, B, F)$ capaz de reconocer cadenas de paréntesis balanceados. $Q_M = \{a, b, c, d\}$ es el conjunto de estados de la máquina; $\Sigma_M = \{(\,,)\}$ es el alfabeto de entrada; $\Gamma = \{(\,,)X,B\}$ es el alfabeto de la cinta, que incluye el alfabeto de entrada y el símbolo de blanco; δ es la función de transición mostrada en el Cuadro 6.1; $q_0 = a$ es el estado inicial de la máquina; B es el símbolo en blanco y $F = \{d\}$ es el conjunto de estados finales.

State	(	)	X	B
a	a,(R	b,X,L	,a,R	c,B,L
b	a,X,R		b,X,L	
c			c,X,L	d,B,R
d				

Cuadro 6.1: Función de transición para la máquina de Turing para el emparejamiento de paréntesis

Podemos apreciar también el diagrama de transiciones para esta máquina en la siguiente figura. Como podemos ver, una vez que la máquina entra en el estado d, no puede volver a cambiar de estado, así que no importa en realidad si se mueve a la derecha o a la izquierda.

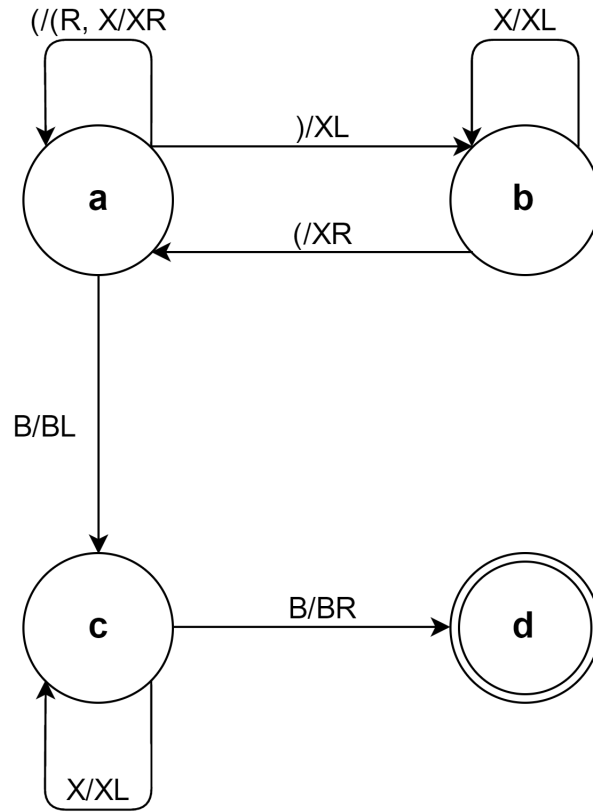


Figura 6.1: Diagrama de transiciones de la máquina de Turing para el emparejamiento de paréntesis

Lo que esta máquina hace al inicio de su ejecución es buscar el primer paréntesis de cierre en la cadena (de izquierda a derecha) y lo cambia por un símbolo X. A continuación busca su correspondiente paréntesis de apertura para cambiarlo por una X también y vuelve a buscar un paréntesis de cierre. La máquina borra todos los paréntesis y, finalmente, cuando encuentra un símbolo en blanco, cambia al estado de aceptación. Si hay paréntesis no balanceados, la máquina encuentra un símbolo en blanco pero no alcanza el estado final. Podemos ver un ejemplo del proceso que ejecuta esta máquina cuando lee como entrada la cadena $((()))()$. Usamos las descripciones instantáneas de la máquina para mostrar su funcionamiento.

$a((())) \vdash (a(())) \vdash ((a())) \vdash (((a))) \vdash ((b(X))) \vdash ((XaX)) \vdash ((XXa)) \vdash$
 $((XbXX)) \vdash ((bXXX)) \vdash (b(XXX)) \vdash (XaXXX) \vdash (XXaXX) \vdash (XXXaX) \vdash$
 $(XXXXa()) \vdash (XXXXX(a)) \vdash (XXXXXb(X)) \vdash (XXXXXXaX) \vdash (XXXXXXXa) \vdash (XXXXXXbXX) \vdash$
 $(XXXXXbXXXX) \vdash (XXXXbXXXXX) \vdash (XXbXXXXXX) \vdash (XbXXXXXXXX) \vdash (bXXXXXXXXX) \vdash$
 $b(XXXXXXXXXX) \vdash XaXXXXXXXXX \vdash XXaXXXXXXXXX \vdash XXXaXXXXXXXXX \vdash XXXXaXXXXXXXXX \vdash$
 $XXXXXaXXXXX \vdash XXXXXXXaXXX \vdash XXXXXXXXaX \vdash XXXXXXXXXaB \vdash XXXXXXXXXcX \vdash$
 $XXXXXXXXcXXBB \vdash XXXXXcXXX \vdash XXXXcXXXXX \vdash XXXcXXXXXXXX \vdash XxcXXXXXXXXX \vdash$
 $XcXXXXXXXXX \vdash cXXXXXXXXX \vdash cBXXXXXXXXX \vdash dXXXXXXXXX$

6.1.2. Representación de la cinta de la máquina de Turing

El autómata construido en esta sección está diseñado no solo para aceptar el mismo lenguaje que la máquina de Turing (el lenguaje de las cadenas de paréntesis balanceados), sino también

para emular su comportamiento. Con este objetivo, estudiamos la dinámica de las descripciones instantáneas mostrándolas en una lista en la que la configuración de la máquina en el paso de tiempo $t+1$ se encuentra inmediatamente debajo de la configuración en el tiempo t , creando una representación bidimensional de la evolución del sistema. Por ejemplo, en el Cuadro 6.2, se muestra esta representación para la cadena del ejemplo de la sección anterior. También se muestra la imagen producida por la herramienta de Software desarrollada en la figura siguiente. Este es el comportamiento que trataremos de emular con el autómata celular.

B	a	(	(	(	)	)	(	)	)
B	(	a	(	(	)	)	(	)	)
B	(	(	a	(	)	)	(	)	)
B	(	(	(	a	)	)	(	)	)
B	(	(	b	(	X	)	(	)	)
B	(	(	X	a	X	)	(	)	)
B	(	(	X	X	a	)	(	)	)
B	(	(	X	b	X	X	(	)	)
B	(	(	b	X	X	X	(	)	)
B	(	b	(	X	X	X	(	)	)
B	(	X	a	X	X	X	(	)	)
B	(	X	X	a	X	X	(	)	)
B	(	X	X	X	a	X	(	)	)
B	(	X	X	X	X	a	(	)	)
B	(	X	X	X	X	(	a	)	)
B	(	X	X	X	X	b	(	X	)
B	(	X	X	X	X	X	a	X	)
B	(	X	X	X	X	X	X	a	)
B	(	X	X	X	X	X	b	X	X
B	(	X	X	X	X	b	X	X	X
B	(	X	X	X	b	X	X	X	X
B	(	X	X	b	X	X	X	X	X
B	(	b	X	X	X	X	X	X	X
B	b	(	X	X	X	X	X	X	X
B	X	a	X	X	X	X	X	X	X
B	X	X	a	X	X	X	X	X	X
B	X	X	X	a	X	X	X	X	X
B	X	X	X	X	a	X	X	X	X
B	X	X	X	X	X	a	X	X	X
B	X	X	X	X	X	X	a	X	X
B	X	X	X	X	X	X	X	a	X
B	X	X	X	X	X	X	X	X	a
B	X	X	X	X	X	X	X	c	X
B	X	X	X	X	X	X	c	X	X
B	X	X	X	X	c	X	X	X	X
B	X	X	c	X	X	X	X	X	X
B	X	c	X	X	X	X	X	X	X
B	c	X	X	X	X	X	X	X	X
c	B	X	X	X	X	X	X	X	X
B	d	X	X	X	X	X	X	X	X

Cuadro 6.2: Lista de descripciones instantáneas al analizar la cadena $((()))()$

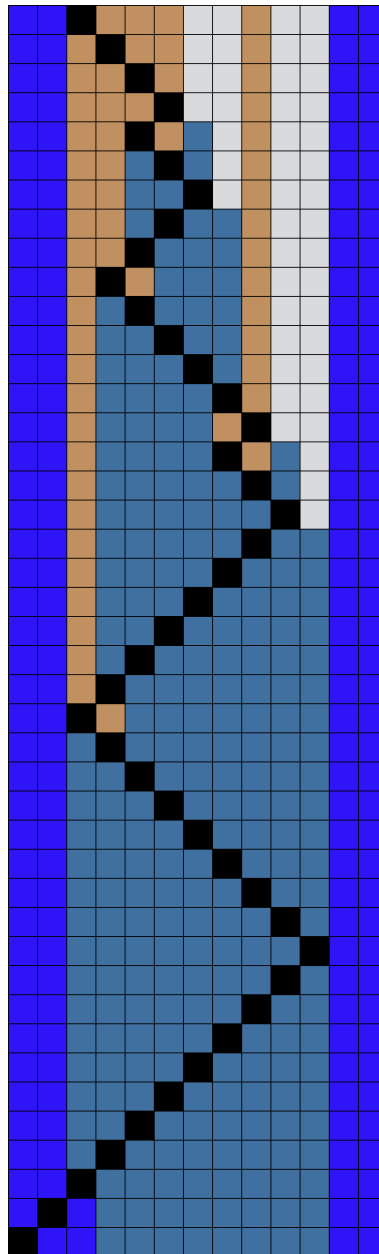


Figura 6.2: Dinámica en dos dimensiones de la máquina para la cadena de entrada $((()))()$

Esto podemos lograrlo haciendo uso de una sola dimensión en el arreglo del autómata celular. Dicho arreglo representa la cinta de la máquina, pero si representamos únicamente la cinta, la evolución en paralelo del autómata celular no nos permite emular el comportamiento de la máquina de Turing tal y como lo hemos mostrado, por lo que tenemos que incluir el cabezal de la misma a la representación tal y como sucede en las descripciones instantáneas.

Recordemos que la cinta de la máquina de Turing es infinita tanto a la izquierda como a la derecha. Para representarla en el autómata celular tenemos dos opciones:

- Representar la cinta con un arreglo infinito tanto a la derecha como a la izquierda lleno de blancos más allá de la cadena que se está analizando tal y como en la cinta. Esto nos permite emular el comportamiento de la máquina aún si esta no para.

- Representar la cinta con un arreglo en el que se agregan los símbolos en blanco suficientes (es decir, que el cabezal de la máquina nunca podrá alcanzar símbolos más allá de los considerados) a cada lado de la cadena que se analiza y conectar sus extremos formando un anillo. Esta representación no nos sirve si el autómata no para, es decir, sólo sería capaz de reconocer un lenguaje decidable. Como el lenguaje en cuestión es decidable, usaremos esta representación.

6.1.3. Construcción del autómata

Con esta información, podemos proponer una definición para el autómata como se describe a continuación. Sea A el autómata celular equivalente, $A = \langle L, \Sigma_A, u, f \rangle$ en donde L es el arreglo previamente descrito, $\Sigma_A = \Gamma \cup Q_M$ (usamos la unión de estos dos conjuntos porque en el autómata representaremos tanto el cabezal de la máquina como la cadena que se analiza). De hecho, necesitaremos más símbolos, pero hablaremos de los mismos más tarde. u es una vecindad de radio, $r = 1$, es decir que la vecindad para la celda x_i será $x_{i-1}x_ix_{i+1}$. Finalmente, la función de transición será descrita en las siguientes subsecciones.

6.1.4. Representación de los movimientos a la derecha

Como lo establecimos previamente, el arreglo del autómata es igual a las descripciones instantáneas de la máquina. La inclusión del cabezal de la máquina permite al sistema procesar (leer y modificar) un solo símbolo a la vez. Recordemos que el cabezal apunta al símbolo que se encuentra a su derecha, por lo que representar las transiciones de la máquina de Turing que producen movimientos del cabezal a la derecha es muy sencillo.

Para simular dichos movimientos, que ocurren con transiciones (de la máquina de Turing) de la forma $\delta\langle p, Y \rangle = \langle q, Y', R \rangle$ en donde R indica que el cabezal se mueve a la derecha, podemos definir fácilmente dos transiciones para el autómata celular. Estas transiciones son $f\langle Z, p, Y \rangle = Y'$ y $f\langle p, Y, Z \rangle = q$, en donde Z es un símbolo arbitrario en Σ_A . Podemos observar esto con claridad en el Cuadro 6.3.

Z	p	Y	→	Y'
p	Y	Z	→	q

Cuadro 6.3: Transiciones del autómata celular que simulan el movimiento de la máquina de Turing a la derecha

Podemos ver un ejemplo de esta representación para la transición $\delta\langle a, (\rangle = \langle a, (, R \rangle$ en el Cuadro 6.4. Podemos observar también cómo funciona para la misma cadena que usamos para mostrar el comportamiento de la máquina de Turing, $((()))()$ en el Cuadro 5. Nótese que las celdas que no se encuentran inmediatamente después del símbolo que representa el cabezal de la máquina permanecen estables. También podemos observar que el comportamiento del autómata celular para estas transiciones es idéntico al comportamiento de la máquina de Turing.

Z	a	(	→	(
a	(	Z	→	a

Cuadro 6.4: Transiciones del autómata celular que simulan la transición $\delta\langle a, (\rangle = \langle a, (, R \rangle$ de la máquina de Turing

a	(	(	(	)	)	(	)	)
(	a	(	(	)	)	(	)	)

Cuadro 6.5: Evolución del autómata celular para los movimientos a la derecha

6.1.5. Representación de los movimientos a la izquierda

Determinar las transiciones del autómata celular correspondientes a las transiciones de la máquina de Turing que hacen que el cabezal se mueva hacia la derecha fue sencillo, pues en esta representación el cabezal lee el símbolo que se encuentra a su derecha, así que simplemente tuvimos que intercambiar los símbolos (realizando los cambios de estado y de símbolo correspondientes). Pero para las transiciones de la máquina que mueven el cabezal a la izquierda es más complicado.

El problema es que no podemos simplemente mover el símbolo que representa el cabezal hacia la izquierda, pues la vecindad del símbolo a la izquierda de este no puede identificar qué símbolo es el que se está leyendo, por lo que es imposible determinar cuál será el siguiente estado. Para solucionar esto, empleamos un símbolo auxiliar.

Usemos la transición $\delta\langle a, \rangle = \langle b, X, L \rangle$ como ejemplo. La primera transición que tenemos que definir para f (la función de transición del autómata celular) es $f\langle Z, a, \rangle = a'$. Nótese que tenemos que agregar a' a Σ_A . Si hay más transiciones en la máquina de Turing que producen un movimiento hacia la izquierda en el estado a , tenemos que añadir un símbolo auxiliar por cada una. Podemos denotar estos símbolos de la forma a'', a^3, a^4, a^5 , etc.

Haciendo uso de estos símbolos auxiliares podemos saber cuál es el siguiente estado de la máquina independientemente de si se puede o no leer el símbolo al que apunta. Finalmente tenemos que añadir otras tres transiciones a f . Dos para mover el cabezal hacia la izquierda y otra para reemplazar el símbolo que se lee. Podemos observar cómo se hace esto en el Cuadro 6.6. También podemos observar el comportamiento del autómata haciendo uso de estas reglas en el Cuadro 6.7. Nótese que es necesario hacer uso de un paso de tiempo adicional para completar la simulación del movimiento a la izquierda.

Z	a	)	→	a'
a'	)	Z	→	X
Z	a'	)	→	Z
Z ₁	Z ₂	a'	→	b

Cuadro 6.6: Transiciones del autómata celular para simular la transición $\delta\langle a, \rangle = \langle b, X, L \rangle$ de la máquina de Turing

(	(	(	a	)	)	(	)	)
(	(	(	a'	)	)	(	)	)
(	(	b	(	X	)	(	)	)

Cuadro 6.7: Evolución del autómata celular con movimientos a la izquierda

6.1.6. Descripción completa de la función de transición del autómata celular

Haciendo uso de las técnicas descritas con anterioridad, podemos construir la función de transición completa para el autómata celular equivalente a la máquina de Turing. El Cuadro 6.8 muestra la lista de transiciones para el autómata (las transiciones que no aparecen permanecen constantes) y el Cuadro 6.9 muestra su comportamiento para la cadena $((()))()$. Como podemos ver, el comportamiento del autómata es casi el mismo que el de la máquina de Turing, sólo que hay algunos pasos extra (aparecen en el autómata pero no en la máquina de Turing) marcados con una flecha a la izquierda de la tabla.

u(x)			f(u(x))	TM's corresponding transition
Z	a	(	(	$\delta\langle a, (\rangle = \langle a, (, R \rangle$
a	(	Z	a	
Z	a	)	a'	$\delta\langle a,) \rangle = \langle b, X, L \rangle$
a'	)	Z	X	
Z	a'	)	Z	
Z ₁	Z ₂	a'	b	
Z	b	(	X	$\delta\langle b, (\rangle = \langle a, X, R \rangle$
b	(	Z	a	
Z	a	X	X	$\delta\langle a, X \rangle = \langle a, X, R \rangle$
a	X	Z	a	
Z	b	X	b'	$\delta\langle b, X \rangle = \langle b, X, L \rangle$
b'	X	Z	X	
Z	b'	)	Z	
Z ₁	Z ₂	b'	b	
Z	a	B	a''	$\delta\langle a, B \rangle = \langle c, B, L \rangle$
a''	B	Z	B	
Z	a''	B	Z	
Z ₁	Z ₂	a''	c	
Z	c	X	c'	$\delta\langle c, X \rangle = \langle c, X, L \rangle$
c'	X	Z	X	
Z	c'	X	Z	
Z ₁	Z ₂	c'	c	
Z	c	B	B	$\delta\langle c, B \rangle = \langle d, B, R \rangle$
c	B	Z	d	

Cuadro 6.8: Descripción de la función de transición del autómata celular equivalente

	a	(	(	(	)	)	(	)	)
	(	a	(	(	)	)	(	)	)
	(	(	a	(	)	)	(	)	)
	(	(	(	a	)	)	(	)	)
→	(	(	(	a'	)	)	(	)	)
	(	(	b	(	X	)	(	)	)
	(	(	X	a	X	)	(	)	)
	(	(	X	X	a	)	(	)	)
→	(	(	X	X	a'	)	(	)	)
	(	(	X	b	X	X	(	)	)
→	(	(	X	b'	X	X	(	)	)
	(	(	b	X	X	X	(	)	)
→	(	(	b'	X	X	X	(	)	)
	(	b	(	X	X	X	(	)	)
	(	X	a	X	X	X	(	)	)
	(	X	X	a	X	X	(	)	)
	(	X	X	X	a	X	(	)	)
	(	X	X	X	X	a	(	)	)
	(	X	X	X	X	(	a	)	)
→	(	X	X	X	X	(	a'	)	)
	(	X	X	X	X	b	(	X	)
	(	X	X	X	X	X	a	X	)
→	(	X	X	X	X	X	X	a'	)
	(	X	X	X	X	X	b	X	X
→	(	X	X	X	X	X	b'	X	X
	(	X	X	X	X	b	X	X	X
→	(	X	X	X	X	b'	X	X	X
...									
	b	(	X	X	X	X	X	X	X
	X	a	X	X	X	X	X	X	X
	X	X	a	X	X	X	X	X	X
	X	X	X	a	X	X	X	X	X
	X	X	X	X	a	X	X	X	X
	X	X	X	X	X	a	X	X	X
	X	X	X	X	X	X	a	X	X
	X	X	X	X	X	X	X	a	X
→	X	X	X	X	X	X	X	X	a''
	X	X	X	X	X	X	X	c	X
	X	X	X	X	X	X	X	c'	X
	X	X	X	X	X	c	X	X	X
→	X	X	X	X	X	c'	X	X	X
	X	X	X	X	X	c	X	X	X
→	X	X	X	X	X	c'	X	X	X
...									
	c	B	X	X	X	X	X	X	X
		d	X	X	X	X	X	X	X

Cuadro 6.9: Evolución del autómata para la cadena $((()))$

A continuación se muestra la ejecución del programa para la máquina de Turing para el emparejamiento de paréntesis y su respectivo autómata celular equivalente obtenido con el algoritmo propuesto con la entrada $((()))((()))((()))$.

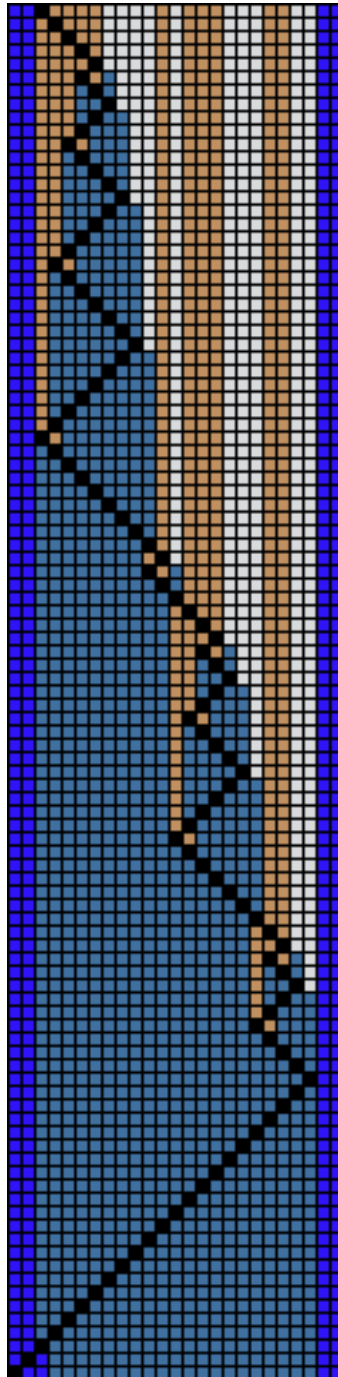


Figura 6.3: Dinámica en dos dimensiones de la máquina para el emparejamiento de paréntesis con la cadena $((()))((()))((()))$

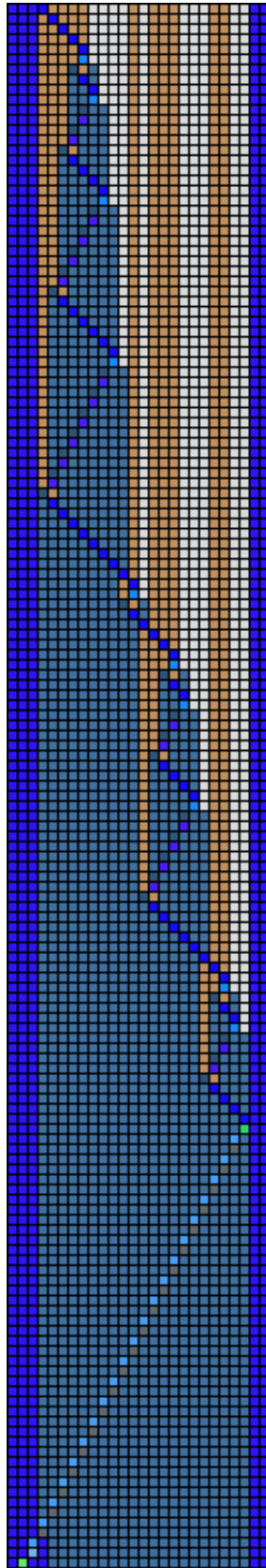


Figura 6.4: Dinámica en dos dimensiones del autómata para el emparejamiento de paréntesis con la cadena $a((((())))(())())$

6.1.7. Condición de paro en el autómata

Sabemos que el autómata ha terminado con la simulación de la máquina de Turing cuando su configuración permanece constante. Es decir que ha terminado de analizar la cadena de entrada w_0 cuando $w_{t+1} = w_t$ en donde w_t es la cadena formada por los símbolos del autómata en el paso de tiempo t .

Decimos que se acepta la cadena w_0 si al finalizar la simulación de la máquina de Turing en el paso de tiempo t , existe un símbolo $Z \in F$ en w_t . De lo contrario, la cadena w_0 no es aceptada.

6.2. Algoritmo para el cálculo de un autómata celular equivalente a una máquina de Turing arbitraria

Haciendo uso de las técnicas utilizadas para calcular un autómata celular equivalente a la máquina de Turing para el reconocimiento del lenguaje de cadenas de paréntesis balanceados que se especificó en el apartado anterior, podemos determinar un algoritmo general que, dada una máquina de Turing arbitraria, calcule un autómata celular equivalente.

Sea M una máquina de Turing arbitraria de la forma $M = (Q, \Sigma_M, \Gamma, \delta, q_0, B, F)$, definimos el autómata celular equivalente A de la forma $A = (L, \Sigma_A, u, f)$, en donde L es un arreglo de celdas unidimensional, u una vecindad de radio $r = 1$, Σ_A un conjunto de estados y f una función de transición de la forma $f : \Sigma_A^u \rightarrow \Sigma_A$. Luego, para calcular cada uno de los elementos del autómata celular, seguimos los siguientes pasos:

- Inicializamos
 - $\Sigma_A = \Sigma_M \cup Q$
 - $f = \emptyset$
- Para cada p, X, q, Y, D tales que $\delta(p, X) = (q, Y, D)$:
 - Si $D = derecha$:
 - Agregar $ZpX \rightarrow Y$ a f
 - Agregar $pXZ \rightarrow q$ a f
 - Si $D = izquierda$:
 - Agregar p^{n+1} como símbolo auxiliar de p a Σ_A , donde n es el número de símbolos auxiliares de p que ya existen
 - Agregar $ZpX \rightarrow p^n$ a f
 - Agregar $p^nXZ \rightarrow Y$ a f
 - Agregar $Zp^nX \rightarrow Z$ a f
 - Agregar $Z_1Z_2p^n \rightarrow q$ a f
- Para todas las demás combinaciones:
 - Agregar $Z_1Z_2Z_3 \rightarrow Z_2$ a f

Figura 6.5: Algoritmo para el cálculo de un autómata celular equivalente a una máquina de Turing arbitraria

6.3. Aplicación del algoritmo

6.3.1. Máquina para la suma binaria

Consideremos la máquina $M = (Q_M, \Sigma_M, \Gamma, \delta, q_0, B, F)$ descrita en [18], capaz de reconocer cadenas de la forma $= a+b =$, en donde a y b son números binarios, y de efectuar su suma. $Q_M = \{a, Q0, dig, zle, ole, zad, oad, car, new, rig, lef, fin\}$ es el conjunto de estados de la máquina, $\Sigma_M = \{=, 1, 0, +\}$ es el alfabeto de entrada, $\Gamma = \{=, 1, 0, +, y, b, B\}$ es el alfabeto de la cinta, δ es la función de transición mostrada en el Cuadro 6.10, $q_0 = a$ es el estado inicial de la máquina y B es el símbolo en blanco.

State	=	1	0	+	B	y	b
a	a,=,R	a,1,R	a,0,R	a,+,R	Q0,B,L		
Q0	dig,B,L						
dig		ole,=,L	zle,=,L	lef,+,L			
zle		zle,1,L	zle,0,L	zad,+,L			
ole		ole,1,L	ole,0,L	oad,+,L			
zad	new,y,L	rig,b,R	rig,y,R			zad,y,L	zad,b,L
oad	new,b,L	car,y,L	rig,B,R			oad,y,L	oad,b,L
car	new,1,L	car,0,L	rig,1,R				
new					rig,=,R		
rig	dig,B,L	rig,1,R	rig,0,R	rig,+,R		rig,y,R	rig,b,R
lef	fin,B,R	lef,1,L	lef,0,L			lef,y,L	lef,b,L
fin		fin,1,R	fin,0,R	halt,B,R		fin,0,R	fin,1,R
halt							

Cuadro 6.10: Función de transición para la máquina de Turing que realiza una suma binaria

Aplicando el algoritmo anteriormente mencionado obtenemos un autómata celular como el que se describe a continuación.

u(x)				f(u(x))	Transición correspondiente de la MT
Z	a	=	=	a	$\delta(a,=) = (a,=,R)$
a	=	Z	a		
Z	a	1	1	a	$\delta(a,1) = (a,1,R)$
a	1	Z	a		
Z	a	0	0	a	$\delta(a,0) = (a,0,R)$
a	0	Z	a		
Z	a	+	+	a	$\delta(a,+) = (a,+,R)$
a	+	Z	a		
Z	a	B	B	Q0	$\delta(a,B) = (q,B,R)$
a	B	Z	Q0		
Z	Q0	=	Q0'	Z	$\delta(Q0,=) = (dig,B,L)$
Q0'	=	Z	B		
Z	Q0'	=	Z		
Z ₁	Z ₂	Q0'	dig		

Z	dig	1	dig'	$\delta(\text{dig},1) = (\text{ole},=,\text{L})$
dig'	1	Z	=	
Z	dig'	1	Z	
Z ₁	Z ₂	dig'	ole	
Z	dig	0	dig''	$\delta(\text{dig},0) = (\text{zle},=,\text{L})$
dig''	0	Z	=	
Z	dig''	0	Z	
Z ₁	Z ₂	dig''	zle	
Z	dig	+	dig ³	$\delta(\text{dig},+) = (\text{lef},+,\text{L})$
dig ³	+	Z	+	
Z	dig ³	+	Z	
Z ₁	Z ₂	dig ³	lef	
Z	zle	1	zle'	$\delta(\text{zle},1) = (\text{zle},1,\text{L})$
zle'	1	Z	1	
Z	zle'	1	Z	
Z ₁	Z ₂	zle'	zle	
Z	zle	0	zle''	$\delta(\text{zle},0) = (\text{zle},0,\text{L})$
zle''	0	Z	0	
Z	zle''	0	Z	
Z ₁	Z ₂	zle''	zle	
Z	zle	+	zle ³	$\delta(\text{zle},+) = (\text{zad},+,\text{L})$
zle ³	+	Z	+	
Z	zle ³	+	Z	
Z ₁	Z ₂	zle ³	zad	
Z	ole	1	ole'	$\delta(\text{ole},1) = (\text{ole},1,\text{L})$
ole'	1	Z	1	
Z	ole'	1	Z	
Z ₁	Z ₂	ole'	ole	
Z	ole	0	ole''	$\delta(\text{ole},0) = (\text{ole},0,\text{L})$
ole''	0	Z	0	
Z	ole''	0	Z	
Z ₁	Z ₂	ole''	ole	
Z	ole	+	ole ³	$\delta(\text{ole},+) = (\text{oad},+,\text{L})$
ole ³	+	Z	+	
Z	ole ³	+	Z	
Z ₁	Z ₂	ole ³	oad	
Z	zad	=	zad'	$\delta(\text{zad},=) = (\text{new},\text{y},\text{L})$
zad'	=	Z	y	
Z	zad'	=	Z	
Z ₁	Z ₂	zad'	new	
Z	zad	1	B	$\delta(\text{zad},1) = (\text{rig},\text{B},\text{R})$
zad	1	Z	rig	
Z	zad	0	y	$\delta(\text{zad},0) = (\text{rig},\text{y},\text{R})$

zad	0	Z	rig	
Z	zad	1	zad''	$\delta(\text{zad},y) = (\text{zad},y,L)$
zad''	y	Z	y	
Z	zad''	y	Z	
Z ₁	Z ₂	zad''	zad	
Z	zad	1	zad ³	$\delta(\text{zad},b) = (\text{zad},b,L)$
zad ³	b	Z	b	
Z	zad ³	b	Z	
Z ₁	Z ₂	zad ³	zad	
Z	oad	=	oad'	$\delta(\text{oad},=) = (\text{new},b,L)$
oad'	=	Z	b	
Z	oad'	=	Z	
Z ₁	Z ₂	oad'	new	
Z	oad	1	oad''	$\delta(\text{oad},1) = (\text{car},y,L)$
oad''	1	Z	y	
Z	oad''	1	Z	
Z ₁	Z ₂	oad''	car	
Z	oad	0	B	$\delta(\text{oad},0) = (\text{rig},B,R)$
oad	0	Z	rig	
Z	oad	y	oad ³	$\delta(\text{oad},y) = (\text{oad},y,L)$
oad ³	y	Z	y	
Z	oad ³	y	Z	
Z ₁	Z ₂	oad ³	oad	
Z	oad	b	oad ⁴	$\delta(\text{oad},b) = (\text{oad},b,L)$
oad ⁴	b	Z	b	
Z	oad ⁴	b	Z	
Z ₁	Z ₂	oad ⁴	oad	
Z	car	=	car'	$\delta(\text{car},=) = (\text{new},1,L)$
car'	=	Z	1	
Z	car'	=	Z	
Z ₁	Z ₂	car'	new	
Z	car	1	car''	$\delta(\text{car},1) = (\text{car},0,L)$
car''	1	Z	0	
Z	car''	1	Z	
Z ₁	Z ₂	car''	car	
Z	car	0	1	$\delta(\text{car},0) = (\text{rig},1,R)$
car	0	Z	rig	
Z	new	B	=	$\delta(\text{new},B) = (\text{rig},=,R)$
new	B	Z	rig	
Z	rig	=	rig'	$\delta(\text{rig},=) = (\text{dig},B,L)$
rig'	=	Z	B	
Z	rig'	=	Z	
Z ₁	Z ₂	rig'	dig	

Z	rig	1	1	$\delta(\text{rig},1) = (\text{rig},1,\text{R})$
rig	1	Z	rig	
Z	rig	0	0	$\delta(\text{rig},0) = (\text{rig},0,\text{R})$
rig	0	Z	rig	
Z	rig	+	+	$\delta(\text{rig},+) = (\text{rig},+,\text{R})$
rig	+	Z	rig	
Z	rig	y	y	$\delta(\text{rig},y) = (\text{rig},y,\text{R})$
rig	y	Z	rig	
Z	rig	b	b	$\delta(\text{rig},b) = (\text{rig},b,\text{R})$
rig	b	Z	rig	
Z	lef	=	B	$\delta(\text{lef},=) = (\text{fin},\text{B},\text{R})$
lef	=	Z	fin	
Z	lef	1	lef'	$\delta(\text{lef},1) = (\text{lef},1,\text{L})$
lef'	1	Z	1	
Z	lef'	1	Z	
Z ₁	Z ₂	lef'	lef	
Z	lef	0	lef''	$\delta(\text{lef},0) = (\text{lef},0,\text{L})$
lef''	0	Z	0	
Z	lef''	0	Z	
Z ₁	Z ₂	lef''	lef	
Z	lef	y	lef ³	$\delta(\text{lef},y) = (\text{lef},y,\text{L})$
lef ³	y	Z	y	
Z	lef ³	y	Z	
Z ₁	Z ₂	lef ³	lef	
Z	lef	b	lef ⁴	$\delta(\text{lef},b) = (\text{lef},b,\text{L})$
lef ⁴	b	Z	b	
Z	lef ⁴	1	Z	
Z ₁	Z ₂	lef ⁴	lef	
Z	fin	1	1	$\delta(\text{fin},1) = (\text{fin},1,\text{R})$
fin	1	Z	fin	
Z	fin	0	0	$\delta(\text{fin},0) = (\text{fin},0,\text{R})$
fin	0	Z	fin	
Z	fin	+	B	$\delta(\text{fin},+) = (\text{halt},\text{B},\text{R})$
fin	+	Z	halt	
Z	fin	y	0	$\delta(\text{fin},y) = (\text{fin},0,\text{R})$
fin	y	Z	fin	
Z	fin	b	1	$\delta(\text{fin},b) = (\text{fin},1,\text{R})$
fin	b	Z	fin	

Cuadro 6.12: Descripción de la función de transición del autómata celular equivalente a la máquina que realiza una suma binaria

A continuación se muestra la ejecución del programa para la máquina de Turing para la suma binaria y su respectivo autómata celular equivalente obtenido con el algoritmo propuesto con la entrada $= 011100 + 010000 =$.

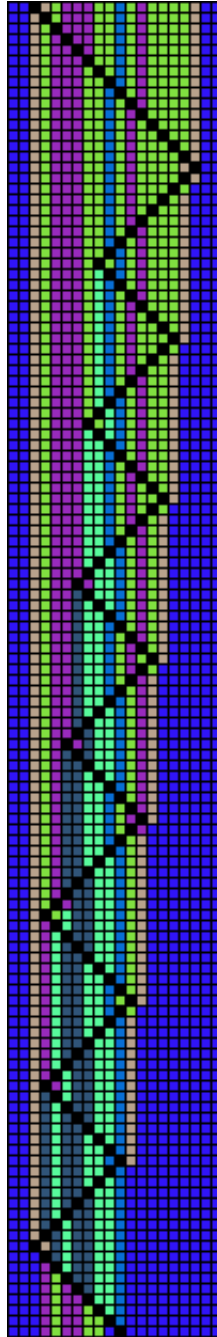


Figura 6.6: Dinámica en dos dimensiones de la máquina para el emparejamiento de paréntesis con la cadena $= 011100 + 010000 =$

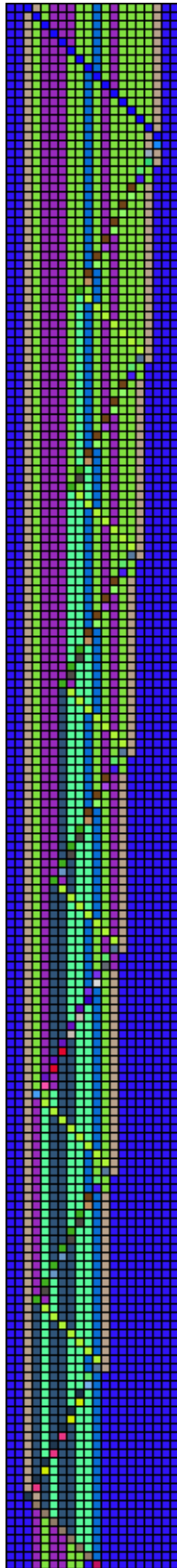


Figura 6.7: Dinámica en dos dimensiones del autómata para el emparejamiento de paréntesis con la cadena $a = 011100 + 010000 =$

6.3.2. Máquina que simula la regla 110

Consideremos la máquina $M = (Q_M, \Sigma_M, \Gamma, \delta, q_0, B)$ descrita en [9], capaz de emular el comportamiento del autómata celular elemental conocido como Regla 110. $Q_M = \{S_{x0}, S_{01}, S_{11}, S_B\}$ es el conjunto de estados de la máquina, $\Sigma_M = \{0, 1\}$ es el alfabeto de entrada, $\Gamma = \{0, 1, B\}$ es el alfabeto de la cinta, δ es la función de transición mostrada en el Cuadro 6.13, $q_0 = S_{x0}$ es el estado inicial de la máquina y B es el símbolo en blanco.

State	0	1	B
S_{x0}	$S_{x0}, 0, R$	$S_{01}, 1, R$	$S_B, 0, L$
S_{01}	$S_{x0}, 1, R$	$S_{11}, 1, R$	
S_{11}	$S_{x0}, 1, R$	$S_{11}, 0, R$	
S_B	$S_B, 0, L$	$S_B, 1, L$	$S_{x0}, 0, R$

Cuadro 6.13: Función de transición para la máquina de Turing que simula la regla 110

Aplicando el algoritmo anteriormente mencionado obtenemos un autómata celular como el que se describe a continuación.

u(x)			f(u(x))	Transición correspondiente de la MT
Z	S_{x0}	0	0	$\delta(S_{x0}, 0) = (S_{x0}, 0, R)$
S_{x0}	0	Z	S_{x0}	
Z	S_{x0}	1	1	$\delta(S_{x0}, 1) = (S_{01}, 1, R)$
S_{x0}	1	Z	S_{01}	
Z	S_{x0}	B	S_{x0}'	$\delta(S_{x0}, B) = (S_B, 0, L)$
S_{x0}'	B	Z	0	
Z	S_{x0}'	B	Z	
Z_1	Z_2	S_{x0}'	S_B	
Z	S_{01}	0	1	$\delta(S_{01}, 0) = (S_{x0}, 1, R)$
S_{01}	0	Z	S_{x0}	
Z	S_{01}	1	1	$\delta(S_{01}, 1) = (S_{11}, 1, R)$
S_{01}	1	Z	S_{11}	
Z	S_{11}	0	1	$\delta(S_{11}, 0) = (S_{x0}, 1, R)$
S_{11}	0	Z	S_{x0}	
Z	S_{11}	1	0	$\delta(S_{11}, 1) = (S_{11}, 0, R)$
S_{11}	1	Z	S_{11}	
Z	S_B	0	S_B'	$\delta(S_B, 0) = (S_B, 0, L)$
S_B'	0	Z	0	
Z	S_B'	0	Z	
Z_1	Z_2	S_B'	S_B	
Z	S_B	1	S_B''	$\delta(S_B, 1) = (S_B, 0, L)$
S_B''	1	Z	1	
Z	S_B''	1	Z	
Z_1	Z_2	S_B''	S_B	
Z	S_B	B	0	$\delta(S_B, B) = (S_{x0}, 0, R)$
S_B	B	Z	S_{x0}	

Cuadro 6.14: Descripción de la función de transición del autómata celular equivalente a la máquina que simula la regla 110

A continuación se muestra la ejecución del programa para la máquina de Turing que simula el funcionamiento de la regla 110 de los autómatas celulares elementales y su respectivo autómata celular equivalente obtenido con el algoritmo propuesto con la entrada 010.

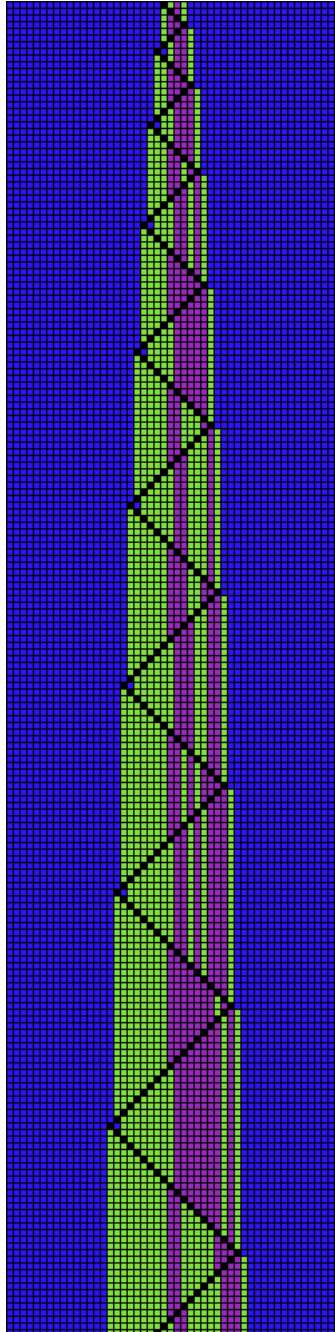


Figura 6.8: Dinámica en dos dimensiones de la máquina que simula el funcionamiento de la regla 110 con la cadena 010

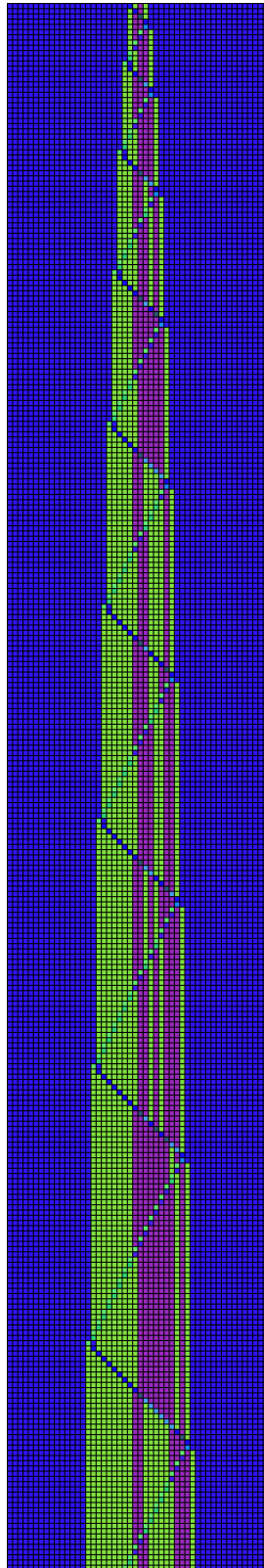


Figura 6.9: Dinámica en dos dimensiones de la máquina que simula el funcionamiento de la regla 110 con la cadena $a010$

6.4. Reducción del número de símbolos auxiliares

Como podemos observar, en el autómata celular que simula la máquina capaz de realizar sumas de números binarios, la cantidad de símbolos auxiliares se dispara a causa de las transiciones a la izquierda, cada una de las cuales necesita de un símbolo adicional. Con el fin de reducir el número de símbolos requeridos para formular un autómata celular universal, debemos efectuar una modificación al algoritmo.

u(x)			f(u(x))	Transición correspondiente de la MT
Z	dig	1	dig'	$\delta(\text{dig}, 1) = (\text{ole}, =, L)$
dig'	1	Z	=	
Z	dig'	1	Z	
Z ₁	Z ₂	dig'	ole	
Z	dig	0	dig''	$\delta(\text{dig}, 0) = (\text{zle}, =, L)$
dig''	0	Z	=	
Z	dig''	0	Z	
Z ₁	Z ₂	dig''	zle	
Z	dig	+	dig ³	$\delta(\text{dig}, +) = (\text{lef}, +, L)$
dig ³	+	Z	+	
Z	dig ³	+	Z	
Z ₁	Z ₂	dig ³	lef	

Cuadro 6.15: Ejemplo de símbolos auxiliares con un comportamiento casi idéntico

Para lograrlo, hemos de considerar partículas que representen el movimiento a la izquierda del cabezal tal y como lo hacen Smith y Lindgren en [19] y [21] respectivamente. Es decir, contaremos con sólo dos partículas para cada estado; una para indicar que el cabezal se desplaza a la derecha y otra para indicar que se desplaza a la izquierda. Ambas partículas serán de tamaño 1. Es decir, cada una será representada con un único símbolo. Pero no basta con reemplazar todos los símbolos auxiliares de un estado por otro símbolo único, pues esto haría que el algoritmo generara reglas ambiguas como se muestra en la siguiente tabla.

u(x)			f(u(x))	Transición correspondiente de la MT
Z	dig	1	dig'	$\delta(\text{dig}, 1) = (\text{ole}, =, L)$
dig'	1	Z	=	
Z	dig'	1	Z	
Z ₁	Z ₂	dig'	ole	
Z	dig	0	dig'	$\delta(\text{dig}, 0) = (\text{zle}, =, L)$
dig'	0	Z	=	
Z	dig'	0	Z	
Z ₁	Z ₂	dig''	zle	
Z	dig	+	dig'	$\delta(\text{dig}, +) = (\text{lef}, +, L)$
dig'	+	Z	+	
Z	dig'	+	Z	
Z ₁	Z ₂	dig'	lef	

Cuadro 6.16: Ejemplo de reglas ambiguas

Para solucionarlo, debemos de modificar la forma en la que se realizan las transiciones a la izquierda. Para ello, durante el primer paso de tiempo, reemplazaremos el símbolo que se lee conforme a lo que indica la regla de la máquina de Turing. De forma simultánea, reemplazaremos el estado, pero indicando que se realizará el movimiento a la izquierda. En el segundo paso de tiempo, el cabezal se moverá a la izquierda, cambiando por el símbolo normal para dicho estado. Así, para una transición de la forma $\delta(p, X) = (q, Y, L)$, definiremos las siguientes transiciones en el autómata celular.

Z	p	X	→	q'
p	X	Z	→	Y

Cuadro 6.17: Reglas para simular una transición de la forma $\delta(p, X) = (q, Y, L)$

Además, para cada estado s tal que $\delta(r, X) = (s, Y, L)$, agregamos la siguiente regla.

Z ₁	Z ₂	s'	→	s
----------------	----------------	----	---	---

Cuadro 6.18: Regla para desplazar el cabezal a la izquierda

Con dichas consideraciones, y tomando como entrada la máquina de Turing que realiza la suma de números binarios, podemos calcular un autómata celular que se comporta de la siguiente manera y que utiliza, a lo más, un símbolo auxiliar por cada estado.

B	a	=	1	1	+	1	0	=	B
B	=	a	1	1	+	1	0	=	B
B	=	1	a	1	+	1	0	=	B
B	=	1	1	a	+	1	0	=	B
B	=	1	1	+	a	1	0	=	B
B	=	1	1	+	1	a	0	=	B
B	=	1	1	+	1	0	a	=	B
B	=	1	1	+	1	0	=	a	B
B	=	1	1	+	1	0	=	Q0'	B
B	=	1	1	+	1	0	Q0	=	B
B	=	1	1	+	1	0	dig'	B	B
B	=	1	1	+	1	dig	0	B	B
...									

Cuadro 6.19: Evolución del autómata celular para los movimientos a la derecha

6.5. Algoritmo para el cálculo de un autómata celular con pocos estados auxiliares equivalente a una máquina de Turing arbitraria

Sea M una máquina de Turing arbitraria de la forma $M = (Q, \Sigma_M, \Gamma, \delta, q_0, B, F)$, definimos el autómata celular equivalente, A , de la forma $A = (L, \Sigma_A, u, f)$, en donde L es un arreglo de celdas unidimensional, u una vecindad de radio $r = 1$, Σ_A un conjunto de estados y f una

función de transición de la forma $f : \Sigma_A^u \rightarrow \Sigma_A$. Luego, para calcular cada uno de los elementos del autómata celular, seguimos los siguientes pasos:

- Inicializamos
 - $\Sigma_A = \Sigma_M \cup Q$
 - $f = \emptyset$
- Para cada p, X, q, Y, D tales que $\delta(p, X) = (q, Y, D)$:
 - Si $D = derecha$:
 - Agregar $ZpX \rightarrow Y$ a f
 - Agregar $pXZ \rightarrow q$ a f
 - Si $D = izquierda$:
 - Si no se ha agregado q' a Σ_A :
 - Agregar q' a Σ_A
 - Agregar $Z_1Z_2q' \rightarrow q$ a f
 - Agregar $pXZ \rightarrow Y$ a f
 - Agregar $ZpX \rightarrow q'$ a f
- Para todas las demás combinaciones:
 - Agregar $Z_1Z_2Z_3 \rightarrow Z_2$ a f

Figura 6.10: Algoritmo para el cálculo de un autómata celular con pocos estados equivalente a una máquina de Turing arbitraria

El autómata celular resultante contará con $\|\Sigma_A\| = \|\Gamma\| + \|Q\| + aux$ estados o colores, en donde aux es el número de estados de la máquina de Turing a los que se puede llegar con un movimiento a la izquierda.

Dado que este algoritmo funciona sin importar las características de la máquina de Turing determinística que toma como entrada, se prueba la primera hipótesis, *Existe un algoritmo tal que, tomando como entrada la descripción formal de una máquina de Turing arbitraria, devuelve como resultado un autómata celular capaz de simular a dicha máquina.*

6.6. Autómata celular universal

6.6.1. Elección de la máquina de Turing universal que se simulará

Utilizando el algoritmo descrito en la sección anterior con una máquina de Turing universal como entrada, se puede calcular un autómata celular universal. Para ello hemos de considerar las máquinas mostradas por Rogozhin en [24], que a su vez aparecen como las máquinas universales más pequeñas en [23]. A continuación se muestra una lista de las máquinas de Yuri Rogozhin. Cada máquina se denota de la forma $UTM(m, n)$ en donde m es el número de estados que posee y n el tamaño del alfabeto de su cinta. Se incluye también el número mínimo y el máximo de estados que podría requerir un autómata celular para simular cada máquina. El mínimo es igual a la suma del número de estados más el tamaño del alfabeto de la cinta de la máquina más uno (pues debe existir al menos una transición a la izquierda), y el máximo es igual a dos veces el número de estados de la máquina menos uno (pues debe de existir al menos una transición a la derecha) más el número de símbolos en el alfabeto.

Máquina	Mínimo	Máximo
UTM(24,2)	27	49
UTM(10,3)	14	25
UTM(7,4)	12	17
UTM(5,5)	11	14
UTM(4,6)	11	13
UTM(3,10)	14	15
UTM(2,18)	21	21

Cuadro 6.20: Máquinas de Yuri Rogozhin

Para determinar a través de qué máquina se puede obtener el autómata celular con menos estados, se calculará el número de auxiliares necesarios para las más pequeñas de estas. Aquellas con un mínimo más grande que el menor máximo, 13, serán descartadas. De esta forma, se analizarán las máquinas $UTM(7, 4)$, $UTM(5, 5)$ y $UTM(4, 6)$. A continuación se muestra las tablas de transiciones de dichas máquinas. Aparecen en negritas aquellas transiciones que agreguen un estado auxiliar.

State	0	1	b	c
q ₁	q₁,0,L	q ₁ ,0,L	q ₂ ,c,R	q ₁ ,b,L
q ₂	q ₂ ,1,R	q ₁ ,0,L	q ₂ ,c,R	q ₅ ,1,R
q ₃	q₄,1,L	q ₃ ,1,R	q ₃ ,c,R	q ₃ ,b,R
q ₄	q₇,1,L	q ₄ ,1,L	q ₄ ,c,L	q ₄ ,b,L
q ₅	q ₄ ,c,L	q ₅ ,1,R	q ₅ ,c,R	q ₅ ,b,R
q ₆	q ₅ ,0,R	q ₆ ,0,R	q ₆ ,b,R	q ₁ ,0,R
q ₇	q ₃ ,0,R		q₆,b,L	

Cuadro 6.21: Función de transición para la máquina $UTM(7, 4)$

State	0	1	b	c	d
q ₁	q ₁ ,1,R	q₁,0,L	q ₁ ,d,R	q ₂ ,0,R	q ₁ ,b,L
q ₂	q ₂ ,0,R	q ₂ ,0,R	q₄,0,L	q ₂ ,c,R	q ₂ ,d,R
q ₃	q ₄ ,c,L	q ₃ ,0,R	q ₅ ,b,R	q ₃ ,c,R	q ₃ ,d,R
q ₄	q ₄ ,1,L	q ₂ ,0,R	q₃,d,L	q ₄ ,c,L	q ₄ ,d,L
q ₅		q ₅ ,1,R		q ₁ ,1,R	q ₅ ,b,R

Cuadro 6.22: Función de transición para la máquina $UTM(5, 5)$

State	1	b	br	bl	0	c
q ₁	q₁,bl,L	q ₁ ,br,R	q ₁ ,b,L	q ₁ ,0,R	q ₁ ,bl,L	q ₄ ,0,R
q ₂	q ₂ ,0,R	q₃,br,L	q ₂ ,bl,R	q₂,br,L	q ₂ ,1,L	q ₂ ,b,R
q ₃	q ₃ ,1,R	q ₄ ,bl,R	q ₃ ,b,R		q ₁ ,c,R	q ₁ ,1,R
q ₄	q ₄ ,0,R	q ₂ ,c,L	q ₄ ,bl,R		q ₂ ,c,L	q ₄ ,b,R

Cuadro 6.23: Función de transición para la máquina $UTM(4, 6)$

Como puede observarse, existen dos máquinas que proporcionan el menor número de estados posibles para el autómata celular, $UTM(5, 5)$ y $UTM(4, 6)$. Se utilizará la máquina

$UTM(4, 6)$ que, además de proporcionar el menor de 13 estados para el autómata celular, tiene una transición menos que la máquina $UTM(5, 5)$.

Máquina	Auxiliares	Total
UTM(7,4)	4	15
UTM(5,5)	3	13
UTM(4,6)	3	13

Cuadro 6.24: Estados necesarios para simular cada máquina

6.6.2. Construcción del autómata celular

Aplicando el algoritmo descrito con anterioridad (Algoritmo para el cálculo de un autómata celular con pocos estados auxiliares equivalente a una máquina de Turing arbitraria), se puede obtener un autómata celular computacionalmente universal de 13 estados a partir de la simulación de la máquina de 4 estados y 6 símbolos descrita por Yurii Rogozhin. Este autómata, A , se describe de la siguiente manera. $A = (L, \Sigma_A, u, f)$, en donde L es un arreglo unidimensional con infinitos símbolos de 0 (símbolo de blanco) tanto hacia la izquierda como la derecha; $\Sigma_A = \{q_1, q_2, q_3, q_4, 1, b, br, bl, 0, c, q_1', q_2', q_3'\}$; u es una vecindad de radio $r = 1$, es decir que contiene 3 celdas incluyendo la que se evalúa y f es la regla de evolución dada por la siguiente tabla.

u(x)			f(u(x))	Transición correspondiente de la MT
Z_1	Z_2	q_1'	q_1	-
Z_1	Z_2	q_2'	q_2	-
Z_1	Z_2	q_3'	q_3	-
Z	q_1	1	q_1'	$\delta(q_1, 1) = (q_1, bl, L)$
q_1	1	Z	bl	
Z	q_1	b	br	$\delta(q_1, b) = (q_1, br, R)$
q_1	b	Z	q_1	
Z	q_1	br	q_1'	$\delta(q_1, br) = (q_1, b, L)$
q_1	br	Z	b	
Z	q_1	bl	0	$\delta(q_1, bl) = (q_1, 0, R)$
q_1	bl	Z	q_1	
Z	q_1	0	q_1'	$\delta(q_1, 0) = (q_1, bl, L)$
q_1	0	Z	bl	
Z	q_1	c	0	$\delta(q_1, c) = (q_4, 0, R)$
q_1	c	Z	q_4	
Z	q_2	1	0	$\delta(q_2, 1) = (q_2, 0, R)$
q_2	1	Z	q_2	
Z	q_2	b	q_3'	$\delta(q_2, b) = (q_3, br, L)$
q_2	b	Z	br	
Z	q_2	br	bl	$\delta(q_2, br) = (q_2, bl, R)$
q_2	br	Z	q_2	

Z	q ₂	bl	q ₂ '	$\delta(q_2, bl) = (q_2, br, L)$
q ₂	bl	Z	br	
Z	q ₂	0	q ₂ '	$\delta(q_2, 0) = (q_2, 1, L)$
q ₂	0	Z	1	
Z	q ₂	c	b	$\delta(q_2, c) = (q_2, b, R)$
q ₂	c	Z	q ₂	
Z	q ₃	1	1	$\delta(q_3, 1) = (q_3, 1, R)$
q ₃	1	Z	q ₃	
Z	q ₃	b	bl	$\delta(q_3, b) = (q_4, bl, R)$
q ₃	b	Z	q ₄	
Z	q ₃	br	b	$\delta(q_3, br) = (q_3, b, R)$
q ₃	br	Z	q ₃	
Z	q ₃	0	c	$\delta(q_3, 0) = (q_1, c, R)$
q ₃	0	Z	q ₁	
Z	q ₃	c	1	$\delta(q_3, c) = (q_1, 1, R)$
q ₃	c	Z	q ₁	
Z	q ₄	1	0	$\delta(q_4, 1) = (q_4, 0, R)$
q ₄	1	Z	q ₄	
Z	q ₄	b	q ₄ '	$\delta(q_4, b) = (q_4, c, L)$
q ₄	b	Z	c	
Z	q ₄	br	bl	$\delta(q_4, br) = (q_4, bl, R)$
q ₄	br	Z	q ₄	
Z	q ₄	0	q ₂ '	$\delta(q_4, 0) = (q_2, c, L)$
q ₄	0	Z	c	
Z	q ₄	c	b	$\delta(q_4, c) = (q_4, b, R)$
q ₄	c	Z	q ₄	

Cuadro 6.26: Descripción de la función de transición del autómata celular equivalente a la máquina que simula la regla 110

Con el cálculo de este autómata celular, se prueba la segunda hipótesis, *Puede obtenerse un autómata celular universal a través de la aplicación del algoritmo anteriormente mencionado a una máquina de Turing universal.*

Capítulo 7

Esbozo de una taxonomía inicial de las máquinas de Turing

Cuando Wolfram propuso su clasificación para los autómatas celulares era sabido que no es posible realizarla analizando todas las configuraciones iniciales posibles sobre todos los tamaños posibles, ya que resultan en un número infinito de combinaciones. Es por ello que el análisis realizado en este apartado será puramente estadístico. Sólo se trabajará con un número limitado de máquinas de Turing, con un número limitado de configuraciones iniciales, un número limitado de tamaños y un número limitado de posiciones de inicio para el cabezal de la máquina. Todo esto sin olvidar que sólo es un bosquejo inicial que requerirá como trabajo a futuro el mismo análisis con más máquinas de Turing.

7.1. El conjunto de máquinas de Turing a analizar

Utilizando el software desarrollado es posible analizar cualquier máquina de Turing dada su descripción formal. Sin embargo, al existir un número ilimitado de máquinas de Turing, sólo fueron ingresadas algunas de ellas. Estas máquinas son casos clásicos y algunas de recientes estudio. Al ejecutar las máquinas en un amplio número de ejecuciones, se asegura que la morfología del espacio es posible ser observada. Éstas fueron:

1. 0^n1^n . Reconoce cadenas de n números cero seguidas por n números uno.
2. Emparejamiento de paréntesis. Reconoce cadenas bien formadas de pares de paréntesis.
3. Duplicador de unos. Dada una cadena de números uno, se duplica de forma unaria la cantidad de números uno.
4. Suma. Dadas dos cadenas binarias y siguiendo un formato específico, se obtiene la suma binaria de estas cadenas.
5. Resta unaria. Dada una cadena de la forma 0^n10^m , siendo 0^n la concatenación de n números cero y 0^m la concatenación de m números cero; se obtiene una cantidad de números cero igual a $n - m$.
6. UTM(7,4). Implementación de una máquina de Turing con 7 estados y 4 símbolos. [\[24\]](#)

Estas máquinas serán nuestra muestra con los diferentes parámetros de variación antes mencionados.

7.1.1. La máquina 0^n1^n

Esta máquina suele llegar a una condición de paro rápidamente en condiciones iniciales aleatorias. Es posible concluir esto como resultado de un experimento en el se ejecuta la máquina con una cinta de entrada aleatoria de 5000 caracteres, con el cabezal iniciando en una posición aleatoria de la cinta de entrada; se obtuvo que de 1000 ejecuciones. 497 (49.7 %) de ellas paran de inmediato y se detienen en la primer generación y 739 (73.9 %) paran a lo más en la quinta generación.

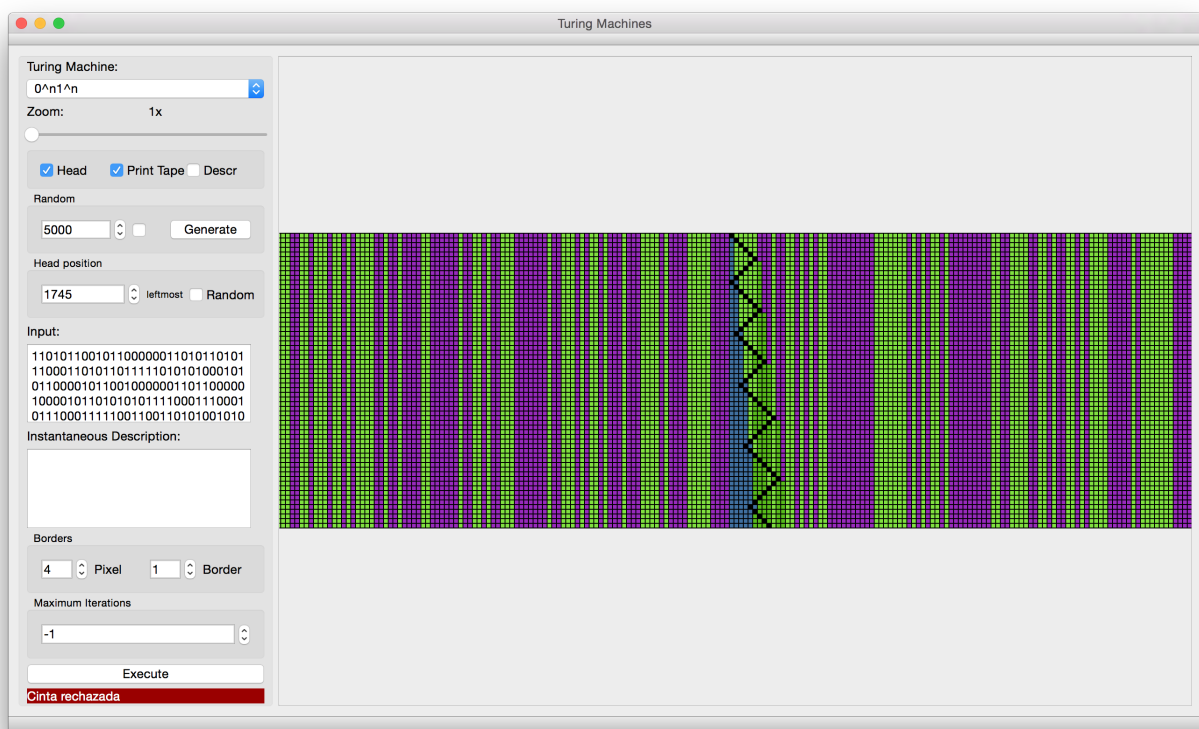


Figura 7.1: Ejecución de la máquina 0^n1^n con un tamaño inicial de 5000 caracteres y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 63 generaciones.

Mientras que en el caso de una cadena que será aceptada por la máquina, que inicia con el cabezal en el extremo izquierdo de la cinta, con longitud 100; alcanza las 20202 generaciones, en comparación con los pocos caracteres que se alcanzan de forma aleatoria.

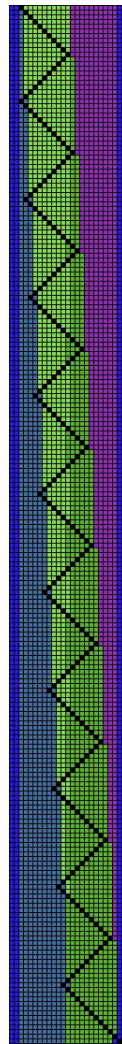


Figura 7.2: Ejecución de la máquina 0^n1^n con un tamaño inicial de 10 caracteres y con posición inicial del cabezal a la izquierda de la cinta de entrada. Esta cinta inicial alcanzó las 222 generaciones.

7.1.2. La máquina para el emparejamiento de paréntesis

Para el análisis de esta máquina se realizó un experimento en el que se ejecutó 1000 veces la máquina de Turing con una cadena inicial aleatoria, con signos de entrada válidos, de longitud de 300 caracteres; ya que fue imposible procesar una cadena más larga con el poder computacional utilizado durante el experimento, debido al número de generaciones alcanzadas.

Durante el experimento se ejecutó la máquina con una posición inicial del cabezal aleatoria y se obtuvo que el promedio de generaciones alcanzadas fue 458. Además, 680 de estas cintas iniciales alcanzó menos de 100 generaciones y el máximo de generaciones alcanzadas fue de 17046.

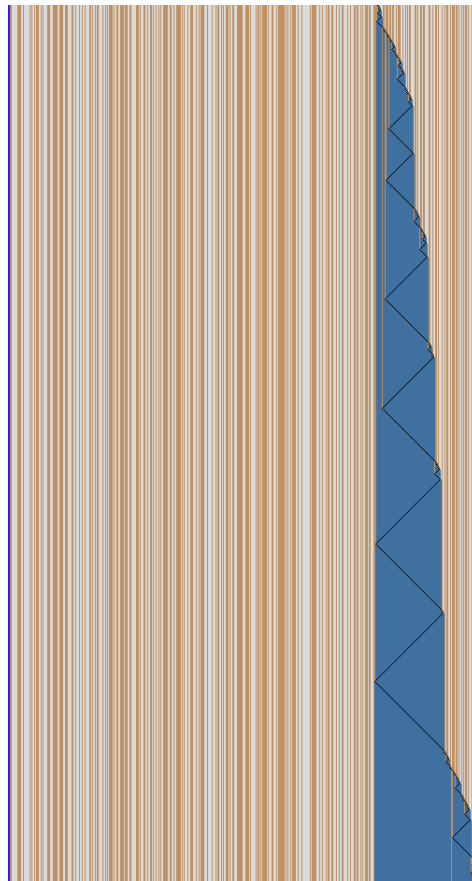


Figura 7.3: Ejecución de la máquina para el emparejamiento de paréntesis con un tamaño inicial de 300 caracteres y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 938 generaciones.

Sin embargo una cadena típica aceptada de longitud de 300 caracteres, que inicia con el cabezal en el extremo izquierdo de la cinta, alcanza fácilmente las 10000 generaciones. En este caso vemos que hay ocasiones en las que una cadena inicial aleatoria es capaz de superar, en número de generaciones alcanzadas, a un caso aceptado, aunque sea sólo en algunos casos.

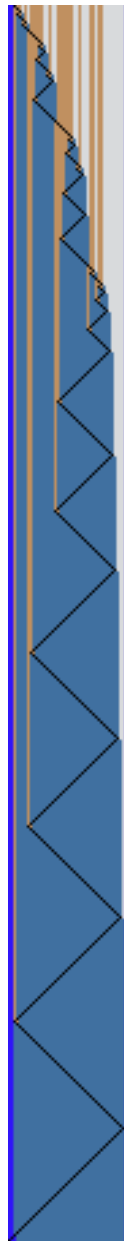


Figura 7.4: Ejecución de la máquina para el emparejamiento de paréntesis con una cadena inicial de 20 caracteres y con posición inicial del cabezal en el extremo izquierdo de la cinta de entrada. Esta cinta inicial alcanzó las 463 generaciones (más que en promedio de forma aleatoria)

7.1.3. La máquina duplicadora de unos

Esta máquina logra llegar a su condición de paro de forma muy precepitada cuando se le dan cadenas iniciales aleatorias e inicia con el cabezal en una posición aleatoria sobre la cadena de entrada; a comparación de las cadenas que son aceptadas con la misma misma longitud. Este resultado fue hallado ejecutando 1000 veces la máquina con cintas aleatorias de longitud 10000 con el cabezal en una posición inicial aleatoria.

Se obtuvo como resultado que en promedio se llegaba a lo más a la cuarta generación.

Además, 751 (75.1 %) de las muestras alcanzaba a lo más 5 generaciones y 931 (93.1 %) alcanzaba a lo más 10 generaciones. El mayor número de generaciones alcanzadas fue de 67.

Empero, una simple cinta que será aceptada de longitud 20, alcanza las 233 generaciones, iniciando con el cabezal a la izquierda de la cadena de entrada.

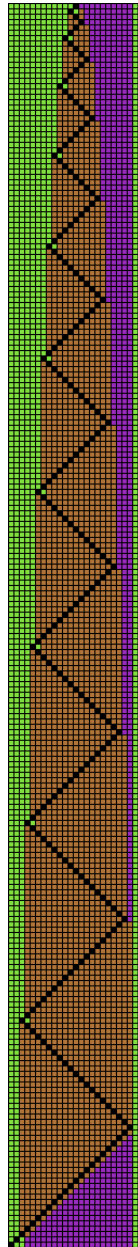


Figura 7.5: Ejecución de la máquina duplicadora de unos con una cadena inicial de 20 caracteres y con posición inicial del cabezal en el extremo izquierdo de la cinta de entrada. Esta cinta inicial alcanzó las 233 generaciones (más de el triple que el máximo obtenido en el experimento)

7.1.4. La máquina sumadora

Al realizar el experimento con la máquina sumadora, ingresando cintas iniciales aleatorias de tamaño 1000 y el cabezal en una posición aleatoria; se obtuvo que el promedio de genera-

ciones alcanzadas fue de 502. Lo cual indica que en la mayoría de los casos la máquina para inmediatamente después de recorrerse al extremo derecho de la cadena de entrada, restricción requerida en la descripción de la misma. Esto provoca que en la mayoría de las ejecuciones aleatorias el comportamiento de la máquina se mantenga estático, aunque con el cabezal desplazándose. Además, 497 (49.7 %) de las muestras sólo llegan a al menos 500 generaciones y 895 (89.5 %) alcanzan las 900 generaciones, lo cual permitiría apenas recorrer el cabezal al extremo derecho de la cadena de entrada en el caso de iniciar en el extremo izquierdo.

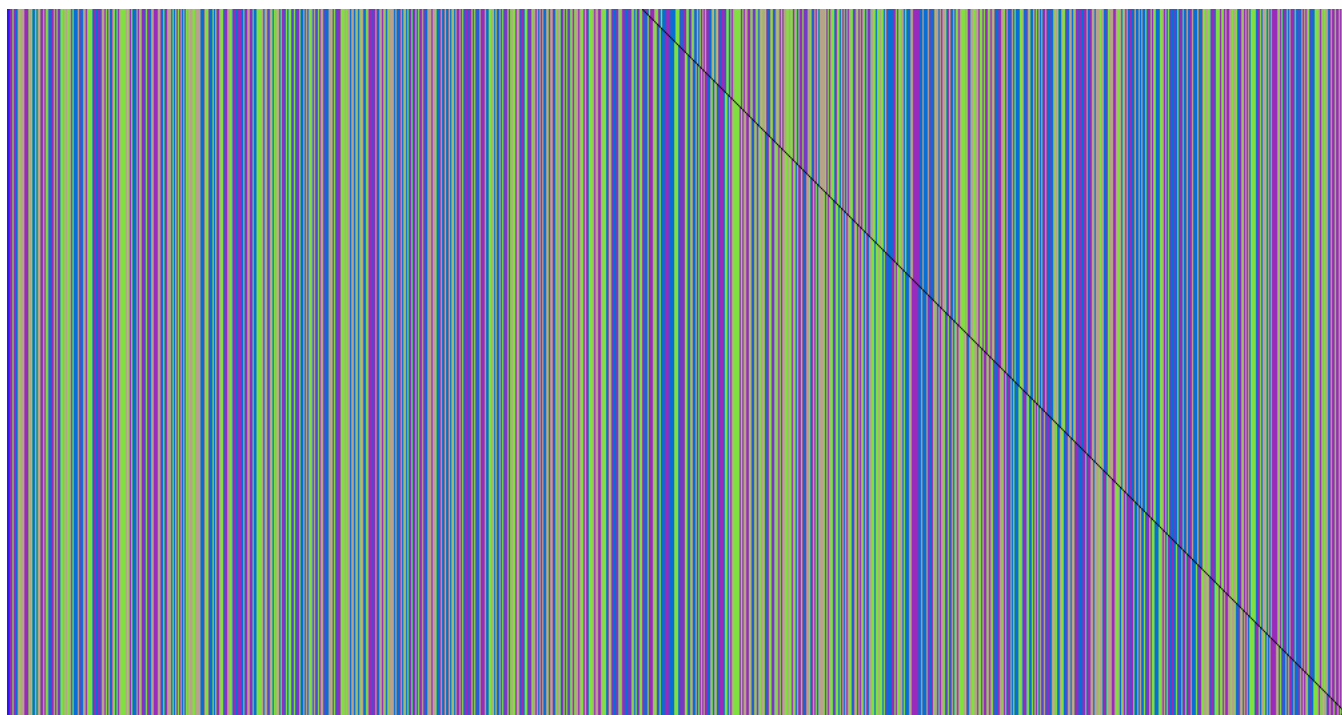


Figura 7.6: Ejecución de la máquina sumadora con una cadena inicial de 1000 caracteres aleatorios y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 1001 generaciones.

Aunado a esto, una cadena aceptada de 1000 caracteres con la que se probó para saber cuánto debería de ser el esperado ese caso, arrojó un total de 504017 generaciones; lo cual resulta mucho más de lo obtenido en los casos aleatorios. Esto puede tener origen en el formato estricto que exige la máquina como entrada para operar y obtener una salida aceptada por la misma.

7.1.5. La máquina para la resta unaria

Para conocer el comportamiento general de esta máquina de Turing, se realizó un experimento ejecutando la máquina con cintas iniciales aleatorias de tamaño 1000; con el cabezal iniciando en una posición aleatoria, obteniendo como resultado que el promedio de generaciones alcanzadas fue de 532. Además, se obtuvo que 469 (46.9 %) muestras alcanzaron menos de 500 generaciones y 862 (86.2 %) obtuvo menos de 900 generaciones. El máximo número de generaciones alcanzadas fue de 1061.

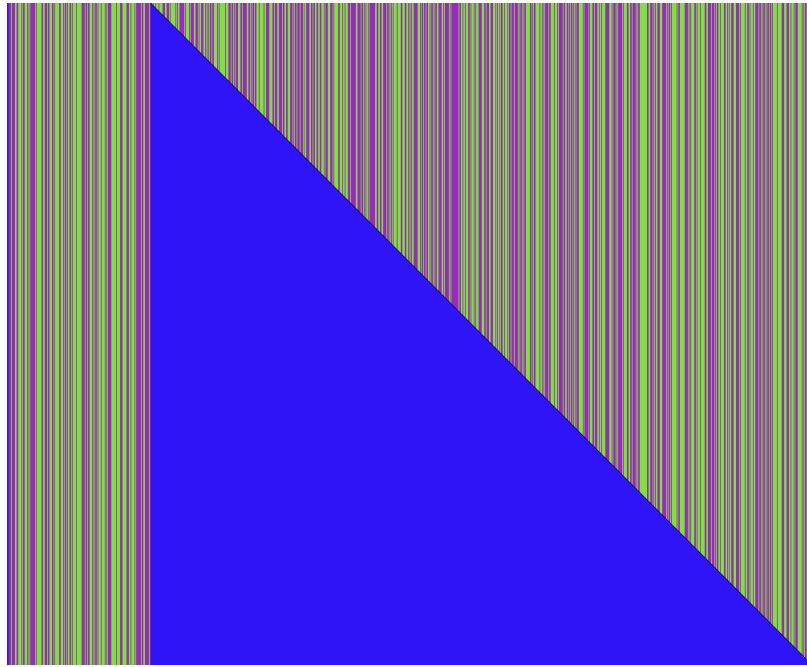


Figura 7.7: Ejecución de la máquina para la resta unaria con una cadena inicial de 1000 caracteres aleatorios y con posición inicial del cabezal aleatoria. Esta cinta inicial alcanzó las 825 generaciones.

Es importante recalcar sobre esta máquina el que se observa un patrón modificando la cinta y no se estaciona de inmediato. Además, para tener una comparación, una cinta de entrada diseñada para alcanzar el mayor número de iteraciones de tamaño 31 llega hasta las 513 generaciones.

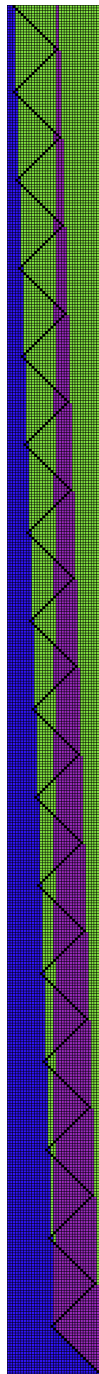


Figura 7.8: Ejecución de la máquina para la resta unaria con una cadena inicial 31 caracteres arbitrarios el cabezal a la izquierda de la cadena de entrada. Esta cinta inicial alcanzó las 513 generaciones.

7.1.6. La máquina de Turing universal con 7 estados y 4 símbolos

Finalmente, trabajamos con esta máquina universal, haciendo el mismo experimento con parámetros similares; con la diferencia de que al ser una máquina universal, está diseñada para no parar. Es por ello que se debe poner un límite de iteraciones. El experimento fue realizado con una cinta inicial de 1000 caracteres, con el cabezal en una posición aleatoria y un límite de

iteraciones de 1000.

Es difícil obtener datos sobre el número de generaciones alcanzadas, ya que sólo se ve detenida por el máximo de iteraciones. Sin embargo, es posible observar que aunque hay movimiento del cabezal, realmente la dinámica está estacionada.

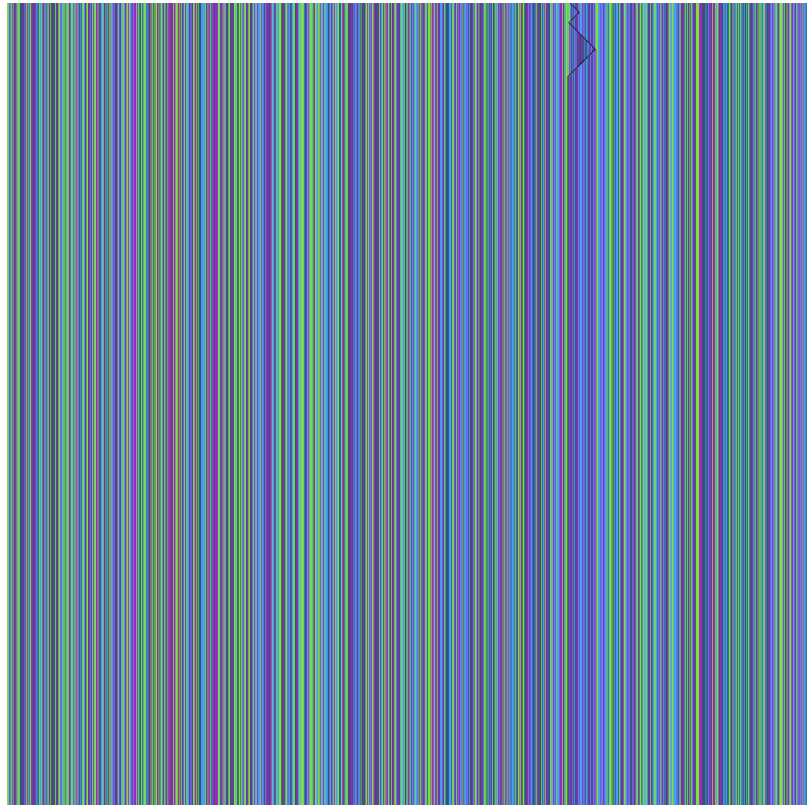


Figura 7.9: Ejecución de la máquina de Turing universal con 7 estados y 4 símbolos con una cadena inicial 1000 caracteres con el cabezal en una posición inicial aleatoria sobre la cadena de entrada.

7.2. Esbozo de la taxonomía

Una vez analizadas las máquinas es posible llegar a varias conclusiones sobre el esbozo de la taxonomía y sobre el trabajo a futuro.

Con base en las máquinas analizadas concluimos la existencia de 3 clases:

- Clase I: Son aquellas máquinas que se estacionan de forma inmediata. Aunque hay movimiento del cabezal o se alcanzan múltiples generaciones, la información de la cinta no se ve realmente modificada. Tal es el caso de la máquina duplicadora de unos, la máquina de $0^n 1^n$, la máquina sumadora o la máquina universal con 7 estados y 4 símbolos.
- Clase II: Son aquellas máquinas que no se estacionan de inmediato, pero cambian la información con algún patrón cíclico. Este es el caso de la máquina para la resta unaria.
- Clase III: Son aquellas máquinas que además de no estacionarse de inmediato, cambian la información con patrones no predecibles, como se puede apreciar en la máquina para

el emparejamiento del paréntesis.

Sin embargo, es importante recalcar que como trabajo a futuro se deja el explorar la dinámica de más máquinas, así como el considerar evoluciones con más una máquina corriendo sobre una misma cinta, además de que esto permitiría considerar el choque de múltiples cabezales corriendo sobre la misma cinta.

Capítulo 8

Conclusiones

Bibliografía

- [1] M. Mitchell, *Complexity, a guided tour*. New York: Oxford University Press, 2009.
- [2] M. Giunti and C. Mazzola. “Dynamical systems on monoids: toward a general theory of deterministic systems and motion” in *Methods, models, simulations and approaches towards general theory of change*. Fifth national conference on Systems Science. World Scientific Publishing Co., pp. 173-185 , 2012.
- [3] R. A. Holmgren. *A first course in discrete dynamical systems*. Springer, 1996.
- [4] A. Ilachinski. *Cellular Automata, a Discrete Universe*. World Scientific, 2001.
- [5] A. Adamatzky. *Computing in Nonlinear Media and Automata Collectives*. Bristol and Philadelphia: Institute of Physics Publishing, 2001.
- [6] S. Wolfram. *A new kind of Science*. Wolfram Media, Inc., 2002.
- [7] C. R. Dyer, “One-Way Bounded Cellular Automata”, *Information and Control*, vol. 44, issue 3, pp. 261-281, March 1980.
- [8] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora and S. V. C. Vergara, “Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases fi 1”, *Journal of Cellular Automata*, vol. 6, issue 2-3, pp. 121-161, 2011.
- [9] M. Cook, “Universality in Elementary Cellular Automata”, *Complex Systems*, vol. 15, issue 1, pp. 1-40, 2004.
- [10] E. W. Weisstein, MathWorld—A Wolfram Web Resource, “Elementary Cellular Automaton”, May 2017. [Online]. Available: <http://mathworld.wolfram.com/ElementaryCellularAutomaton.html>
- [11] E. W. Weisstein, MathWorld—A Wolfram Web Resource, “Rule 110”, May 2017. [Online]. Available: <http://mathworld.wolfram.com/Rule110.html>
- [12] Wolfram Research, *Mathematica*, 2017.
- [13] A. Wuensche. *DDLab*, May 2017. [Online]. Available: <http://www.ddlab.com/>
- [14] J. E. Hopcroft, R. Motwani and J. D. Ullman *Teoría de Autómatas, Lenguajes y Computación*, Pearson Educación S.A., 2007.
- [15] C. E. Shannon, “A universal Turing machine with two internal states”, in *Annals of Mathematics Studies - Automata Studies*, pp. 157-165, 1956.

-
- [16] M. Tamayo and Tamayo. *El proceso de la Investigación Científica. Incluye evaluación y administración de proyectos de investigación*. Ed. Limusa Noriega Editores, 2003.
- [17] M. Fourment and M. R. Gillings, “A comparison of common programming languages used in bioinformatics”, *BMC Bioinformatics*, vol. 9, issue 82, February 2008.
- [18] C. Gerardo, *REC and Convert as aids in teaching Automata Theory*. Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias Universidad Autónoma de Puebla.
- [19] A. R. Smith *Simple Computation-Universal Cellular Spaces*. New York university, The Bornx, New York. 1971.
- [20] A. Jürgen *A Simple Universal Cellular Automaton and its One-Way and Totalistic Version*. Complex Systems Publications, Inc. 1987.
- [21] K. Lindgren *Universal Computation un Simple One-Dimensional Cellular Automata*. Complex Systems Publications, Inc. 1990.
- [22] S. Wolfram. *Wolfram Alpha*, Oct 2017. [Online]. Available: <https://www.wolframalpha.com/>
- [23] D. Woods *The complexity of small universal Turing machines: A survey*. Theoretical Computer Science. 2009.
- [24] Y. Rogozhin *Small universal Turing machines*. Elsevier. 1996